VARIATIONAL AUTOENCODER FOR SYNTHETIC INSURANCE DATA

Charlotte Jamotton, Donatien Hainaut

REPRINT | 2024 / 38

UCLouvain

ISBA | LIDAM
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

Intelligent Systems with Applications 24 (2024) 200455

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Variational AutoEncoder for synthetic insurance data

Charlotte Jamotton *, Donatien Hainaut

Institute of Statistics, Biostatistics and Actuarial Sciences (ISBA) of the Université catholique de Louvain (UCLouvain), Voie du Roman Pays 20/L1.04.01, Louvain-la-Neuve, 1348, Belgium

ARTICLE INFO

Keywords:
Autoencoder
Variational inference
Synthetic data generation
Heterogeneous insurance data
Dimension reduction

ABSTRACT

This article explores the application of Variational AutoEncoders (VAEs) to insurance data. Previous research has demonstrated the successful implementation of generative models, especially VAEs, across various domains, such as image recognition, text classification, and recommender systems. However, their application to insurance data, particularly to heterogeneous insurance portfolios with mixed continuous and discrete attributes, remains unexplored. This study introduces novel insights into utilising VAEs for unsupervised learning tasks in actuarial science, including dimension reduction and synthetic data generation. We propose a VAE model with a quantile transformation for continuous (latent) variables, a reconstruction loss that combines categorical cross-entropy and mean squared error, and a KL divergence-based regularisation term. Our VAE model's architecture circumvents the need to pre-train and fine-tune a neural network to encode categorical variables into n-dimensional representative vectors within a continuous vector space of dimension R^n. We assess 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 in addressing challenges related to data availability in the insurance industry.

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. However, insurance data is known to contain sensitive and private information, such as the policyholder's name, address, or gender, subject to regulatory requirements.[1] Merely deleting these identification factors does not guarantee complete anonymisation of the data. Moreover, the more data we conceal, the less valuable information is left for future analysis. This data anonymity issue has sparked interest in synthetic data and its associated deep learning-based generative model.

A generative model, or synthesis, is trained to generate artificial data that share similar structure and characteristics with the original data. Therefore, the authentic and synthetic data should 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 computer-generated data is no longer personally identifiable information or associated with a data subject. Synthetic data could serve as a valuable insurance tool, not only to sidestep privacy regulations that restrict access to actual data but also to handle situations where actual data is unavailable (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 Goodfellow et al. (2014) and the Variational AutoEncoders (VAEs) proposed by Kingma and Welling (2013) are the two most prominent approaches to generative modelling. They have established themselves over the last few years for their ability to generate highly realistic samples of images, texts, or sounds. While GANs are often praised for their ability to generate high-quality visual samples, they are also singled out for their time-consuming iterative optimisation. On the other hand, VAEs are known for their relatively easy implementation and hyper-parameters setup, as well as their explicit inference network and tractable likelihood, but tend to generate blurry or fuzzy samples (see, e.g., Zhao, Song, & Ermon, 2017). The bottleneck architecture, unique to autoencoders, led us to prioritise VAEs over the computationally expensive GANs in this research article. The appeal of a bottleneck layer

* Corresponding author.
E-mail addresses: charlotte.jamotton@uclouvain.be (C. Jamotton), donatien.hainaut@uclouvain.be (D. Hainaut).

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

https://doi.org/10.1016/j.iswa.2024.200455
Received 29 January 2024; Received in revised form 24 September 2024; Accepted 21 October 2024
Available online 29 October 2024

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 human perception, where we focus on core features rather than intricate details (e.g., we do not recollect every aspect of a dog's appearance to recognise or distinguish it from a cat, we focus on core characteristics instead). This hypothesis is the motivation behind dimension reduction techniques such as principal component analysis (Abdi & Williams, 2010), t-distributed Stochastic Neighbour Embedding (Van der Maaten & 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.

In the actuarial literature, the use of Neural Networks (NNs) has 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-organising 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 neural networks' use in supervised actuarial applications. In such supervised approaches, the network learns a function $f : X \to Y$ where X is 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 optimisation problem where we aim to learn and understand the structure of our inputs rather than trying to map inputs to target(s).

This research article sets out 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 actuarial 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 multi-modal 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 for the categorical attributes is needed) with a loss function that combines a reconstruction loss, as a mixture of categorical cross-entropy and mean squared error, and a regularisation term defined by KL divergence.
4. The generation of high-quality synthetic insurance policies with a quantile transformation of the latent space.

The article is organised as follows. Section 2 provides a thorough state-of-the-art analysis of related studies. Section 3 introduces the VAE model with a theoretical review of the classical (variational) autoencoder and motivates its use as a generative process. Section 4 introduces our main contributions and adjustments to the classical VAE. Section 5 explains our experiment settings and results. Section 5.1 introduces the heterogeneous insurance portfolio used to evaluate the quality of our VAE model. Section 5.2 reviews the setup of the model's hyper-parameters, evaluates the reconstructed portfolio's quality and the latent space's structure, and illustrates the impact of introducing the quantile transformation. Section 5.3 investigates whether our VAE can generate high-quality synthetic insurance portfolios characterised by discrete and continuous variables. Section 6 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 have not yet been investigated in the actuarial literature.

2. Related works

Generative modelling from learned latent representations is a central application of Variational AutoEncoders (Kingma & Welling, 2013). This technique plays a crucial role in many sectors, including insurance, where accurate risk assessment, modelling, and decision-making processes require access to high-quality, diverse, and representative data. Over the years, researchers have developed various data augmentation techniques to address this need for more data that closely mimic authentic data characteristics to ensure consistent results under statistical analysis. In natural language processing, Wei and Zou (2019) proposed text augmentation techniques such as replacing and inserting random words in a sentence to generate variations and create new contexts to improve the performance of text classification models. Similarly, Xie, Dai, Hovy, Luong, and Le (2020) used unsupervised data augmentation techniques for text classification, such as back translation and paraphrase generation, to improve the model robustness through consistency training. In computer vision, Krizhevsky, Sutskever, and Hinton (2017) employed simple data augmentation techniques like horizontal flipping, random cropping, and zooming to create mirror images and simulate changes in position to improve the generalisation performance of deep convolutional neural networks (CNNs). Similarly, to expand image datasets applied to CNNs, Simard, Steinkraus, Platt, et al. (2003) discusses elastic distortions, including rotation (to simulate different viewing angles), scaling, and translation. These traditional data augmentation techniques often fail to capture the underlying structure and dependencies within the data. Advanced data synthesis methods, such as generative adversarial networks (Goodfellow et al., 2014) and variational autoencoders (Kingma & Welling, 2013), have shown promise in overcoming these limitations. For a comparative review of deep generative models, including variational autoencoders, generative adversarial networks, normalising flows, and energy-based and auto-regressive models, we refer to Bond-Taylor, Leach, Long, and Willcocks (2021). For a general overview of GANs' most widely recognised variants, their latent architectures, validation metrics, and application areas, we refer to Chakraborty, Ks, Naik, Panja, and Manvitha (2024).

In recent years, several authors have introduced variants of Kingma and Welling (2013) VAE model by developing novel architectures tailored to specific application fields. Real-world applications of VAEs span various domains, including healthcare, finance, natural language processing, and computer vision. In the medical domain, VAEs have been used for disease diagnosis (García-Ordás, Benítez-Andrades, García-Rodríguez, Benavides, & Alaiz-Moretón, 2020; Mansour et al., 2021), abnormal lesions or pathology's detection in brain MR images (An & Cho, 2015; Baur, Wiestler, Albarqouni, & Navab, 2019), patient monitoring (Miotto, Li, Kidd, & Dudley, 2016), drug discovery (Lim, Ryu, Kim, & Kim, 2018; Song et al., 2022), and discrete molecule data generation (Kusner, Paige, & Hernández-Lobato, 2017). In finance, they have been applied to fraud detection (Alazizi, Habrard, Jacquenet, He-Guelton, & Oblé, 2020; Tingfei, Guangquan, & Kuihua, 2020), risk assessment (Mancisidor, Kampffmeyer, Aas, & Jenssen, 2021), and portfolio optimisation (Duan, Wang, Zhang, & Li, 2022; Zhang, Liang, Lyu, & Fang, 2020). In natural language processing, VAEs are used for text generation (Semeniuta, Severyn, & Barth, 2017; Wang et al., 2019; Zhang, Yang, Yuan, Shen, & Carin, 2019), text classification (Xu, Sun, Deng, & Tan, 2017), sentiment analysis (Lu, Zhao, Yin, Yang, & Li, 2020; Wu et al., 2019), recommender systems (Li & She, 2017), and machine translation (Pagnoni, Liu, & Li, 2018). In computer vision, they are employed for image synthesis (Huang, He, Sun, Tan, et al., 2018), object recognition (Li, Sun, & Guo, 2019), and image captioning (Shen, Liu, Zhou, Zhao, & Liu, 2020).

One direction of research that is of interest in actuarial applications involves integrating heterogeneous data in the design of the VAE's architecture. This often entails employing separate encoding pathways for each data type before merging them in the latent space and imposing constraints on the latent space to enforce domain-specific properties.

By integrating domain knowledge, VAEs can learn more meaningful representations and capture relevant dependencies among different data types. Despite significant progress, several challenges remain in the application of VAEs to real-world datasets due to their disparate nature (i.e., (positive) numerical, binary, Multinomial, and ordinal) with a variety of marginal distributions (e.g., multi-modal, long-tail).

The presence of mixed-type values, and more specifically categorical rating factors, has already been addressed in the actuarial literature by Noll et al. (2020) and Ferrario et al. (2020) through the use of entity (Guo & Berkhahn, 2016) or joint (Delong & Kozak, 2021) embedding. However, both approaches require a pre-training of layers to fine-tune numerical representations of categorical features. The time allotted to the model calibration further increases with the number of categorical attributes. This does not align with our intention to train an all-in-one unsupervised autoencoder model.

Instead of mapping categorical variables to continuous vectors, several authors (Antelmi, Ayache, Robert, & Lorenzi, 2019; Nazabal, Olmos, Ghahramani, & Valera, 2020; Öğretir, Ramchandran, Papatheodorou, & Lähdesmäki, 2022) have integrated heterogeneous and incomplete data in the design of VAE-based methods with various likelihood models. Xu and Veeramachaneni (2018) and Xu, Skoularidou, Cuesta-Infante, and Veeramachaneni (2019) foundational papers explore the generation of (medical and educational records) mixed-type tabular data. 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 normalisation using Gaussian mixture models (Bishop & Nasrabadi, 2006). However, for each continuous variable, this normalisation scheme increases the input dimension by the number of Gaussians used to approximate the variable's distribution.

Nazabal et al. (2020) and Ma, Tschiatschek, Turner, Hernández-Lobato, and Zhang (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. Öğretir et al. (2022) took the idea a step further by designing a VAE model for time-series and longitudinal datasets. To capture correlations among different samples, they modelled the time-dependent dynamics with auxiliary covariate information and allowed for various likelihoods to handle heterogeneous data. Antelmi et al. (2019) introduced another extension to tackle the interpretability issues of the latent space. By imposing a constraint on the latent representations, they were able to disentangle the contribution of a single channel in the description of the latent representation and jointly account for latent relationships across heterogeneous data.

Although integrating diverse data types to enhance learning with VAEs has been extensively studied, dealing with multi-modal marginal distributions and the associated mode-seeking behaviour of the KL divergence remains a challenge that has been overlooked in the literature — with the exception of Bishop and Nasrabadi (2006) who proposed the mode-specific normalisation mentioned above. In this article, we propose a quantile transformation of the continuous variables as well as a novel synthesising approach (that also makes use of the quantile transformation) to tackle the collapse of the approximate posterior distribution towards a single mode and the biased generation of samples towards this specific mode. By introducing the quantile transformation at the input and latent space levels, we not only succeeded to capture the multiple modes of the original continuous variables distributions in the reconstructed and synthetic data distributions, but we also improved the deviance performance of a regression model (GLM) trained on the synthetic data to predict the claim frequencies of the original data.

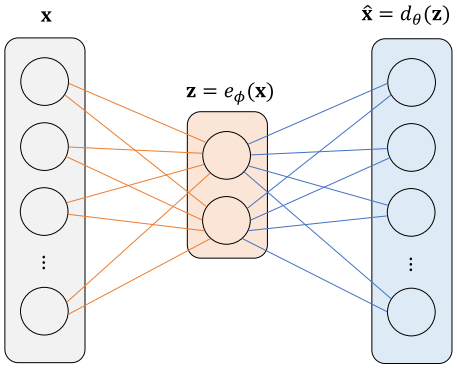


Fig. 1. Architecture of a shallow autoencoder.

3. Variational AutoEncoder

3.1. 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. This encoding contains enough relevant information for the decoder to convert it back to the original input. Formally, the encoder network learns a non-linear transformation,

$$e_\phi : \mathcal{X} \to \mathcal{Z},$$

parameterised by ϕ that maps each input vector $\boldsymbol{x}$ of dimension p from the original high-dimensional input space $\mathcal{X}$ to a lower-dimensional latent space $\mathcal{Z}$. The latent encoding,

$$\boldsymbol{z} = e_\phi(\boldsymbol{x}),$$

is a low-dimensional representation of the original input. Conversely, the decoder network learns a non-linear transformation,

$$d_\theta : \mathcal{Z} \to \mathcal{X},$$

parameterised by θ that projects the latent codes $\boldsymbol{z}$ of dimension $d < p$ back into the original high-dimensional input space $\mathcal{X}$. This transformation takes the latent vector $\boldsymbol{z}$ as input to reconstruct the original input data,

$$\hat{\boldsymbol{x}} = d_\theta(\boldsymbol{z}).$$

The autoencoder,

$$f(\boldsymbol{x}) = d_\theta\left(e_\phi(\boldsymbol{x})\right),$$

illustrated in Fig. 1, is trained as a whole to minimise the difference between each input $\boldsymbol{x}$ and its reconstruction $\hat{\boldsymbol{x}}$.

The traditional encoder network projects each input vector $\boldsymbol{x} \in X$ (e.g., each insurance policy) to a *single* latent value z_j for each encoding dimension $j = 1, \ldots, d$. Therefore, each sample $\boldsymbol{x}$ 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, where policies with similar risk profiles may be mapped to very different or distant latent encodings. It 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:

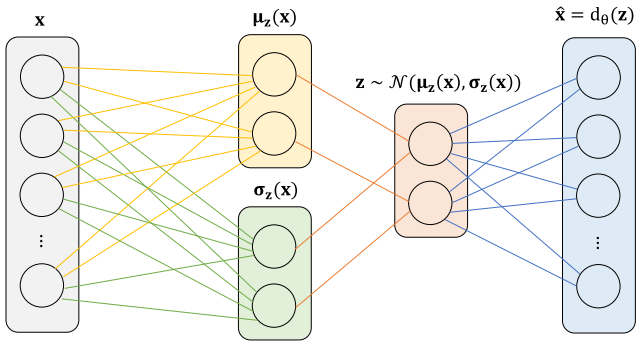


Fig. 2. Architecture of a shallow variational autoencoder with a Gaussian prior.

1. Continuity: the transformation $e_\phi : \mathcal{X} \to \mathcal{Z}$ should ensure that (dis)similar samples $\boldsymbol{x}$ have (dis)similar latent vectors $\boldsymbol{z}$. 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.

Kingma and Welling (2013) and Rezende, Mohamed, and Wierstra (2014) introduced the Variational AutoEncoder (VAE) as a Bayesian extension to the autoencoder (Kramer, 1991) to address this shortcoming. The NNs of their Bayesian model, illustrated in Fig. 2, learn a probabilistic (instead of deterministic) representation of the data points in the latent space. This approach should improve the decoder's robustness to small changes in $\boldsymbol{z}$.

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

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

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

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

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

we have

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

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

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

In practice, 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 $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ 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 maximisation of the log-likelihood lower bound (found by variational inference).

3.2. Variational approximation of the posterior

The exact posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ is approximated with a tractable distribution known as the variational or approximate posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. 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 $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is used to perform approximate inference of the intractable posterior distribution $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. 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 $p_\theta(\boldsymbol{z})$ 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:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z} \mid \boldsymbol{\mu}_z(\boldsymbol{x}), \boldsymbol{\Sigma}_z(\boldsymbol{x})\right) , \tag{5}$$

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

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

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

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

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

3.3. Variational lower bound

Recall that we would like to approximate some posterior distribution $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ with a tractable distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and make this approximate inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ as close as possible to the true, but intractable, posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. Mathematically, we can find such an approximation by deriving a variational lower bound that will make the likelihood function, in Eq. (4), tractable. Such a 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:

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

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

$$\ln\left(p_\theta(\boldsymbol{x})\right) = D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,\|\, p_\theta(\boldsymbol{z}|\boldsymbol{x})\right) + \mathcal{L}(\boldsymbol{x}; \theta, \phi) , \tag{6}$$

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

$$\mathcal{L}(\boldsymbol{x}; \theta, \phi) = \mathbb{E}_{q_\phi}\left[-\ln q_\phi(\boldsymbol{z}|\boldsymbol{x}) + \ln p_\theta(\boldsymbol{x}, \boldsymbol{z})\right] . \tag{7}$$

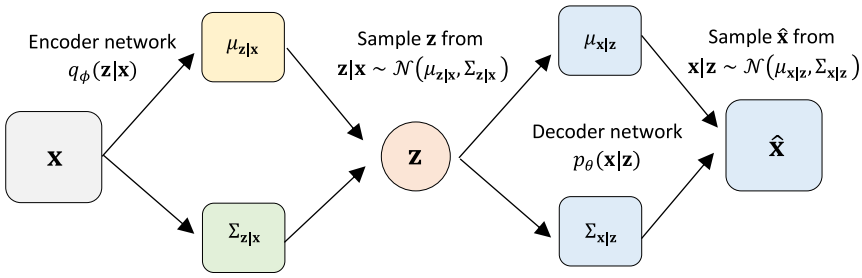


Fig. 3. Illustration of the sampling process for a Gaussian encoder and decoder. The random node is represented by a circle.

The proof (Kingma & Welling, 2013) is given in . The first term in Eq. (6) is the KL divergence of the approximate from the true posterior. It is intractable due to $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. However, since the KL divergence is a measure of the difference between two probability densities, it is always positive. Therefore, the marginal log-likelihood (or evidence) is lower bounded by the variational lower bound $\mathcal{L}$:

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

Therefore, minimising the KL divergence between the exact posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ and the variational $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is equivalent to maximising this variational lower bound $\mathcal{L}$ during the training process. As a result, the complex inference problem is reduced to a simpler optimisation problem:

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

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

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}; \theta, \phi) &= \mathbb{E}_{q_\phi}\left[-\ln q_\phi(\boldsymbol{z}|\boldsymbol{x}) + \ln p_\theta(\boldsymbol{z}) + \ln p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \ln p_\theta(\boldsymbol{z})\right] \\ &= -\mathbb{E}_{q_\phi}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z})}\right] + \mathbb{E}_{q_\phi}\left[\ln \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{p_\theta(\boldsymbol{z})}\right] \\ &= -D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})\right) + \mathbb{E}_{q_\phi}\left[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] . \end{aligned} \tag{8}$$

The first term in Eq. (8) is the KL divergence of the approximate posterior (i.e., the distribution of the encoder outputs) from the prior distribution $p_\theta(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z} \mid 0, \boldsymbol{I})$. 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*

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

where d is the dimension of the latent space.

The proof is given in . The KL divergence term in Eq. (8) may be interpreted as a regulariser or encoding error. It penalises the learned distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ when it diverges too much from the true prior distribution $p_\theta(\boldsymbol{z})$. This penalty term enables the use of the decoder network as a generative model. Because the probabilistic autoencoder model maps each data point onto a distribution instead of a single point in the latent space, the variational encoding covers an area whose centre and size are controlled by the means and standard deviations. However, if the outputted means are significantly different *and* the standard deviations are too small, there may still be some gaps in the latent space, leaving the decodable latent space discontinuous — which is precisely what we want to avoid. By introducing the KL divergence as a regulariser, we force the encoder to find latent vectors that approximately follow a standard Gaussian distribution, encouraging the latent vectors to occupy a more centralised and uniform location. This should help to meet the regularity requirements of the latent space.

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. (8) prevents that by encouraging the decoder's output $\hat{\boldsymbol{x}}$ to reconstruct the original input $\boldsymbol{x}$. Formally, it represents the cross-entropy of $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ w.r.t. q_ϕ:

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

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 parameterise a likelihood distribution that places enough probability mass on the true data. While the KL divergence in Eq. (8) admits an analytical expression (see Proposition 2), the reconstruction loss in Eq. (8) is usually approximated by L simulations:

$$\mathbb{E}_{q_\phi}\left[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] \approx \frac{1}{L}\sum_{l=1}^{L} \ln p_\theta\left(\boldsymbol{x}|\boldsymbol{z}^{(l)}\right) , \tag{9}$$

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

3.4. Reparameterisation trick

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

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

While the error term $\boldsymbol{\varepsilon}$ introduced is stochastic, the node $\boldsymbol{z}$ is now deterministic, enabling the back-propagation through it. Mathematically, we can now optimise $\mathcal{L}$ with respect to the parameters $\phi = \{\boldsymbol{\mu}, \Sigma\}$ of the variational posterior using the back-propagation technique. θ and

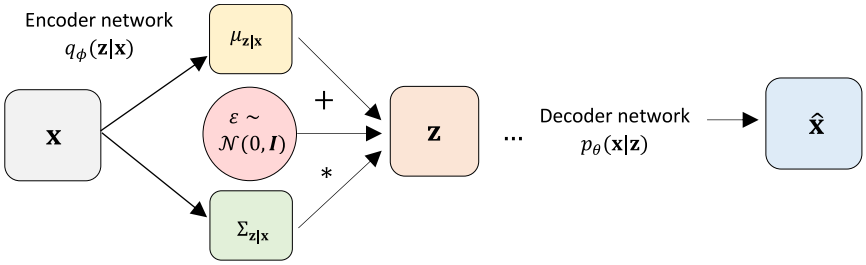


Fig. 4. Illustration of the reparameterisation trick. The random node is represented by a circle.

ϕ are obtained by maximising,

$$\sum_{x\in X} \tilde{\mathcal{L}}(\boldsymbol{x};\theta,\phi) = \sum_{x\in X}\left(-D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})\right) + \frac{1}{L}\sum_{l=1}^{L} \ln p_\theta\left(\boldsymbol{x}|\boldsymbol{z}^{(l)}\right)\right), \tag{10}$$

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

This random generation of $\boldsymbol{z}$ with $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \boldsymbol{I})$ implies the encoding varies, for the same input $\boldsymbol{x}$, in 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 encoding variations for the same input during training. Consequently, the decoder 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. As a result, every point close to the encoded location of one of the input policies will be decoded to something similar to that original policy, thereby making the VAE highly suitable for the generation of new samples. The number of simulations L in the gradient estimator in Eqs. (9) and (10) can also be set to 1 — as the decoder is already exposed to variations (and the mini-batch size m will be set large enough).

4. Heterogeneous data

The VAE model we have just defined was first designed (Kingma & Welling, 2013) with continuous variables in mind, for applications like image generation where the input and output variables are continuous pixel values. This creates challenges for the model application to real-world datasets (e.g., insurance portfolios), which contain various types of features (i.e., (positive) numerical, binary, Multinomial, and ordinal) with a variety of marginal distributions (e.g., multi-modal, long-tail). In the following subsections, we will address and discuss the challenges related to mapping categorical variables to numeric vectors, the multi-modal distributions of continuous variables, and the need for the reconstruction loss to reflect the heterogeneity of the input data. In summary, we propose to handle such heterogeneous datasets by:

1.1. Applying one-hot (or dummy) encoding to the categorical (i.e., binary, ordinal, and Multinomial) variables before inputting them into the encoder network.
1.2. Passing each set of dummy variables belonging to the same categorical variable through a soft-max activation function in the output layer of the decoder network.
2.1. Applying a quantile (non-parametric) transformation to the continuous variables to ensure their distribution follows a uni-modal normal distribution before inputting them into the encoder network.
2.2. Applying a min–max scaling to the range (−1,1) to the normal quantile transformed continuous variables to conform with the range of the tanh activation function used in the output layer of the decoder network.

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

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

3. Introducing a weighted combination of mean squared error and categorical cross-entropy for the reconstruction loss (that is minimised during the training process) to account for the different distributions of the continuous and categorical variables after pre-processing (i.e., Normal and Multinomial distributions).

4.1. One-hot encoding and soft-max activation

Deep learning models require all inputs to be numeric. However, insurance policies are known to also contain information in categorical forms (e.g., geographic location, vehicle power, brand, or colour). Whether nominal (e.g., colour) 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 approaches to represent categorical variables numerically. It maps each modality to a binary vector, as illustrated in Table 1.

Mathematically, after one-hot encoding, each categorical variable X_k with c_k categories will be transformed into c_k binary variables/columns:

$$X_k \to \left\{X_{k1}, X_{k2}, \ldots, X_{kc_k}\right\},$$

where $x_{ikj} \in \{0,1\}$ for $j = 1,2,\ldots,c_k$ and $i = 1,\ldots,n$. Therefore, if X has b categorical variables, and c_k is the number of categories for the kth variable, the total number of columns in the one-hot encoded data, denoted as p_b, is:

$$p_b = \sum_{k=1}^{b} c_k$$

One of the main issues with one-hot encoding is that the dimension of the resulting sparse binary vectors scales linearly with the number of categories present in the data. Additionally, introducing new categories requires retraining the model completely as the new binary vectors and their corresponding input nodes should also be introduced. Moreover, since binary vectors lack semantic significance at the encoder's input layer, the model may struggle to capture the data underlying structure and understand relationships or similarities between different categories based on their one-hot representations (see Table 1).

Entity embedding (Guo & Berkhahn, 2016) has been gaining momentum in the data field for addressing some of the weaknesses of

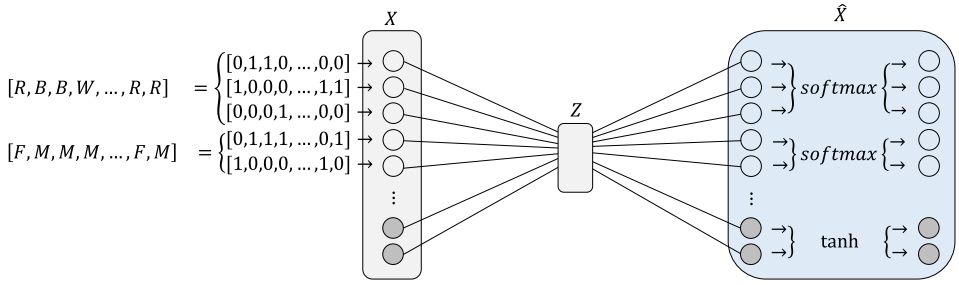


Fig. 5. Example of the slicing technique applied on the decoder output of the Variational AutoEncoder with two categorical variables `Colour` and `Gender` whose categories Red (R), Black (B), White (W), and Female (F), Male (M), are dummy encoded. The darker nodes represent continuous variables (see Section 4.2).

dummy encoding. By learning low-dimensional dense vector representations of categories, embeddings aim to capture semantic relationships between categories and further offer 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 are computationally intensive, and fine-tuning the NNs hyper-parameters can quickly become time-consuming — compared to the ease of implementation of dummy encoding. Furthermore, even entity embedding takes the detour of an ordinal or one-hot encoder before passing the so-encoded input into a set of layers. As pointed out by Richman (2021), the first layer of an autoencoder learns a representation of all of the input attributes that is comparable to an embedding layer. 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.

Vector quantisation via codebooks (Wu & Yu, 2019) is conceptually related to entity embedding, as both approaches involve learning low-dimensional representations of categorical variables. However, they differ in their underlying objectives to address the weaknesses of dummy encoding. While entity embedding overcomes the inability of dummy encoding to capture semantic relationships between categories based on their one-hot representations, vector quantisation via codebooks focuses on the scalability issue of one-hot encoding. By using a fixed set of representative vectors, known as a codebook, the model generalises well to unseen categories by mapping them to the nearest vectors in the codebook and, therefore, does not require retraining the model when new categories are introduced, unlike one-hot encoding.

Although alternative encoding methods like entity embedding or codebook quantisation may enhance learning with VAEs, they add complexity to the model architecture and training procedure. Moreover, these methods do not address at once all the weaknesses associated with one-hot encoding. When considering the trade-offs between simplicity, interpretability, and performance, one-hot encoding is still a reasonable choice for representing categorical variables in machine learning models, especially when the number of categories is relatively small. Furthermore, the interpretability of the binary vectors can be valuable for understanding and analysing the latent space of the VAE.

It is worth mentioning that the choice of the encoding method (one-hot encoding, entity embedding, or vector quantisation) is part of a pre-processing step applied to the data before inputting it into the encoder network of the VAE. Some authors (Jang, Gu, & Poole, 2016; Van Den Oord, Vinyals, et al., 2017) took the idea of handling categorical variables one step further by introducing the Vector Quantised Variational AutoEncoder (VQ-VAE) and Gumbel-softmax trick approach, both of which involve major changes in the latent space assumptions. Given the exclusively categorical nature of their input data, they propose discrete latent representations with categorical posterior and prior distributions instead of the traditional continuous learning representations (with a Gaussian prior). Although the use of discrete latent representations has been shown (Jang et al., 2016; Van Den Oord et al., 2017) to be a more natural fit for many applications inherently discrete (e.g., language, speech, images), there is no theoretical evidence that switching to a categorical latent representation with categorical prior and posterior distributions could supersede the traditional continuous latent representations when dealing with heterogeneous datasets containing both categorical and continuous data.

Our intention to develop a straightforward and all-in-one unsupervised autoencoder led us to select dummy encoding as a baseline approach to map our categorical variables to numeric vectors and to focus on the traditional continuous latent representation (with a Gaussian prior) to study the impact of introducing a quantile transformation (see following Section 4.2) on the reconstructed and synthetic data.

Given the choice of one-hot encoding to transform the categorical variables, we opted for softmax activation functions for the decoder. Because our VAE architecture jointly models the categorical and continuous variables in the input data, the decoder's output layer will be sliced to extract the values corresponding to all modalities of the same categorical variable. Each slice will be individually passed through a softmax activation function, as illustrated in the Fig. 5.

4.2. 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 optimisation by increasing the speed of convergence of the gradient descent and preventing the optimisation from getting stuck in a local optimum (see, e.g., Goodfellow, Bengio, & Courville, 2016; LeCun, Bottou, Orr, & Müller, 2002). Standardisation and normalisation 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, multi-modal, and/or non-normal distributions). The challenges of training NNs on multi-modal distributions have already been highlighted in the literature. Among others, Pascanu, Mikolov, and Bengio (2013) discussed how the vanishing gradient problem is exacerbated by multi-modal 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 generalisation. In summary, if the input distributions are multi-modal 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 given the input data, denoted as $p(\boldsymbol{z}|\boldsymbol{x})$, is also multi-modal (i.e., it has multiple distinct peaks or modes), minimising 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 generated samples being biased towards a specific mode, neglecting the other modes of the true data distribution. This mode-seeking behaviour of the KL divergence can further 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, & Liò, 2020), incorporating adversarial training (Mescheder, Nowozin, & Geiger, 2017), employing more flexible generative models like Normalising Flows (Rezende & Mohamed, 2015), or introducing a mode-specific normalisation (Xu & Veeramachaneni, 2018). The representation of a continuous column C_i after applying this mode-specific normalisation becomes the concatenation of a normalised 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 normalisation further increases the (already significant) input dimension by the number of Gaussians used to *approximate* the multi-modal distributions.

To address these challenges, we propose a quantile transformation to compel continuous variables to follow an uni-modal normal distribution. Compared to the aforementioned techniques, our non-parametric quantile transformer does not require significant changes in the VAE architecture, maintains the numeric input dimension, tends to spread out the most frequent values, smooth out unusual distributions, and is less influenced by outliers than more traditional scaling methods. 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 to 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 normal distribution $\mathcal{N}(\mu,\sigma)$, 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.

Mathematically, we apply the quantile transformation Φ^{-1} to the marginal cumulative distribution function $F(X_j)$ of each continuous rating factor X_j, where $j = 1, 2, \ldots, p_c$ denotes the continuous variable index in the dataset X:

$$X'_j = \Phi^{-1}\left(F(X_j)\right).$$

After applying the quantile transformation, we further apply min–max scaling to each normally transformed variable X'_j. This scaling ensures that the transformed values of each variable fall within the range $(-1,1)$ across all policies. Mathematically, for each x'_{ij}:

$$x''_{ij} = \frac{x'_{ij} - \min(X'_j)}{\max(X'_j) - \min(X'_j)} \times (b_s - b_i) + b_i \qquad \text{for } b_i = -1, b_s = 1,$$

where $\min(X'_j)$ and $\max(X'_j)$ respectively denote the minimum and maximum values of the jth continuous variable across all policies. The range $(-1,1)$ 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 continuous variables before passing them through a tanh activation function (see darker nodes in Fig. 5).

4.3. Heterogeneous reconstruction loss

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

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

During the training, the error calculated by this loss function $\tilde{\mathcal{L}}(\boldsymbol{x};\theta,\phi)$ is propagated backward through the network to update the network and variational parameters (θ,ϕ) and eventually find the optimal values (i.e., the values that minimise the error). Therefore, $\tilde{\mathcal{L}}(\boldsymbol{x};\theta,\phi)$ plays a crucial role in the model's convergence. Its purpose is dual. First, the KL divergence regulariser term D_{KL} encourages the VAE to learn a continuously meaningful latent space, which enables the generation of new meaningful policies. Second, the reconstruction loss $\ln p_\theta\left(\boldsymbol{x}|\boldsymbol{z}^{(l)}\right)$ encourages the VAE to learn a latent space that captures the salient features of the heterogeneous insurance portfolio which enables its reconstruction given its latent representation. In other words, optimising the reconstruction log-likelihood and regulariser simultaneously encourages the latent state to have distributions close to the prior $\mathcal{N}(0,1)$ but deviating when necessary. The β-VAE model proposed by Higgins et al. (2017) offers a way to control the balance between reconstruction accuracy and latent space disentanglement by introducing a hyper-parameter $\beta > 1$ to increase the weight on the KL divergence term. While this hyper-parameter β can easily be introduced into our loss function, choosing an appropriate value for it would introduce additional tuning challenges, and better disentanglement may come at the cost of blurrier reconstructions (Higgins et al., 2017).

While the KL divergence D_{KL} admits an analytical expression (see Proposition 2), the reconstruction loss $\ln p_\theta\left(\boldsymbol{x}|\boldsymbol{z}^{(l)}\right)$ 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 X. Because we assume X to be a real-world dataset containing various types of features with various marginal distributions, the decoder output – and the associated reconstruction loss – should reflect its heterogeneity. Assume the first p_c dimensions of each input sample $\boldsymbol{x} \in X$ contain the observed values of the scaled numeric variables $(X_j)_{j=1,\ldots,p_c}$ and the following $p_b = p - p_c$ dimensions contain the observed values of the b dummy encoded categorical variables $(X_k)_{k=1,\ldots,b}$. After one-hot-encoding, the b categorical variables are transformed into p_b dummy variables, where $p_b = \sum_{k=1}^{b} c_k$ and c_k is the number of modalities taken by the kth categorical variable. After pre-processing, each sample $\boldsymbol{x}$ is described by a set of scaled continuous values and a set of dummy values, such that the insurance dataset X can be written as:

$$X = (\boldsymbol{x}_i)_{i=1,\ldots,n} = \left(\{x_{i,j}\}_{j=1,\ldots,p_c}, \{x_{i,k+p_c}\}_{k=1,\ldots,p_b}\right)_{i=1,\ldots,n}.$$

After pre-processing (normal quantile transform), each continuous variable X_j is assumed to follow a uni-modal Gaussian distribution. Maximising the likelihood of a model whose predictions are assumed to follow a Gaussian distribution is equivalent to minimising mean squared error. In addition, each one of the dummy encoded categorical variables is treated as a multi-class classification problem. Maximising the likelihood of a model whose predictions are assumed to follow a Multinomial distribution is equivalent to minimising 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 $\boldsymbol{x}$ 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),

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

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

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

where

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

and the Scaled Exponential Linear Unit (SELU) (Klambauer, Unterthiner, Mayr, & Hochreiter, 2017) and hyperbolic tangent (tanh) activation functions are respectively defined as:

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

In this case, the probabilistic decoder has density:

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

with a log-likelihood equal to:

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

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

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

where

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

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

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

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

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

where $y(\boldsymbol{z}) = (y_j(\boldsymbol{z}))_{j=1,\ldots,c_k}$ are the outputs of a single layer neural network:

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

where

$$\text{softmax}(x) = \left(\frac{e^{x_i}}{\sum_j^{c_k} e^{x_j}}\right)_{i=1,\ldots,c_k}, \tag{11}$$

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:

$$\ln p(\boldsymbol{x}|\boldsymbol{z}) = \Sigma_{j=1}^{c_k} x_j \ln\left(y_j(\boldsymbol{z})\right). \tag{12}$$

Since the categorical variables $(X_k)_{k=1,\ldots,b}$ are one-hot encoded, there is only one element, denoted $x^{(p)}$, in the target vector $\boldsymbol{x} = \left\{x_1,\ldots,x_{c_k}\right\}$ which is not zero. We can rewrite Eq. (12) as follow:

$$\ln p(\boldsymbol{x}|\boldsymbol{z}) = \ln\left(\frac{e^{x^{(p)}}}{\sum_{j=1}^{c_k} e^{x_j}}\right).$$

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

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 data type (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.

5. Model training

5.1. 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 two modalities), *Zone* (with seven modalities), and *Class* (with seven 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 v_i indicating the contract duration (exposure). After selecting contracts with a duration $v_i \in]0,12]$ months, the dataset counts $n = 62\,289$ policies. As expected for a motor portfolio it is highly imbalanced, with zero claims accounting for 98.95% of the policies, one claim for 1%, and two claims for 0.05%.

The *OwnersAge*, *VehicleAge* and *Duration* variables all follow a multi-modal distribution (see Fig. 6). We use the quantile (non-parametric) transformation, detailed in Algorithm 2, to map the values of each continuous variable to an uni-modal 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. 6.

The categorical variables *Zone*, *Class*, *Gender* and *NumberClaims* are one-hot encoded with, respectively, 7, 7, 2, and 3 dummy columns. A quick analysis of our insurance tabular data reveals (see Fig. 7) 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 locations (Zone 5, 6, and 7) are also strongly under-represented. When dealing with imbalanced data, controlling the model architecture and hyper-parameters setup like the batch size can play an important role in the performance of

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

Algorithm 1: Variational AutoEncoder for heterogeneous insurance portfolio.

Data: Insurance portfolio $X^{(n\times p)}$ consisting of n insurance policies characterised by p_c continuous rating factors and b discrete rating factors:

$$X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}_p = \left\{ \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}_{p_c}, \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}_b \right\}_p$$

Pre-processing step: For $k = 1, \dots, b$ and $j = 1, \dots, p_c$:

$x_{ik} \leftarrow \left\{ x_{ik1}, x_{ik2}, \dots, x_{ikc_k} \right\}$ where $x_{ikj} \in \{0, 1\}$ for $j = 1, 2, \dots, c_k$

$x_{ij} \leftarrow \left\{ x''_{i1}, x''_{i2}, \dots, x''_{ij} \right\}$ where $x''_{ij} = \frac{x'_{ij} - \min(X'_j)}{\max(X'_j) - \min(X'_j)} \times 2 + 1$ and $X'_j = \boldsymbol{\Phi}^{-1}\left(F(X_j)\right)$

Initialisation:

$\alpha, m, N \leftarrow$ Set up hyper-parameters (learning rate α, batch size m, number of epochs N)

$\theta, \phi \leftarrow$ Initialise parameters

while (θ, ϕ) *not converged* **do**

- $X^m \leftarrow$ Draw random mini-batch $X^m = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_m\}$ of $m \le n$ policies
- **for** *each* $\boldsymbol{x} \in X^m$ **do**
  - $\boldsymbol{\epsilon} \leftarrow$ Generate random noise $\boldsymbol{\epsilon} = \{\boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_m\}$ from the auxiliary noise distribution $p(\boldsymbol{\epsilon}) \sim \mathcal{N}(0, \boldsymbol{I})$
  - $\boldsymbol{\mu}, \log\left(\boldsymbol{\sigma}^2\right) \leftarrow e_\phi(\boldsymbol{x})$
  - **for** *each* $l \leftarrow 1$ **to** m **do**
    - $\boldsymbol{z}^{(l)} \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}^{(l)}$
  - Compute continuous $\mathcal{L}^c_{rec}$ and discrete $\mathcal{L}^d_{rec}$ reconstruction loss:
  - $\mathcal{L}^c_{rec} \leftarrow \frac{1}{2m} \sum_{l=1}^m \| \{\boldsymbol{x}\}_{p_c} - f\left(\left\{ d_\theta\left(\boldsymbol{z}^{(l)}\right) \right\}_{p_c} \right) \|^2$ with $f(y) \overset{tanh}{=} \frac{e^y - e^{-y}}{e^y + e^{-y}}$
  - $\mathcal{L}^d_{rec} \leftarrow \frac{1}{m} \sum_{l=1}^m \left(\sum_{k=1}^b \left(\sum_{j=1}^{c_k} \{x_j\}_{c_k} \ln\left(g\left(\left\{ d_\theta\left(\boldsymbol{z}^{(l)}\right) \right\}_{c_k} \right) \right) \right) \right)$ with $g(x) \overset{softmax}{=} \frac{e^x}{\sum_j^{c_k} e^{x_j}}$
  - Compute KL regulariser term:
  - $\mathcal{L}_{KL} \leftarrow -\frac{1}{2m} \sum_{l=1}^m \sum_{j=1}^d \left(1 + \ln \sigma^2_{z,j}(\boldsymbol{x}) - \sigma_{z,j}(\boldsymbol{x})^2 - \mu_{z,j}(\boldsymbol{x})^2 \right)$.
- $\boldsymbol{g} \leftarrow$ Compute the gradients $\nabla_{\theta,\phi} \widetilde{\mathcal{L}}^m(X^m, \epsilon; \theta, \phi)$ of mini-batch estimator w.r.t. (θ, ϕ).
- **update** parameters θ, ϕ using gradients $\boldsymbol{g}$:
  - $\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{\boldsymbol{x} \in X^m} \mathcal{L}_{rec}(\boldsymbol{x})$ with $\mathcal{L}_{rec}(\boldsymbol{x}) = \mathcal{L}^c_{rec}(\boldsymbol{x}) + 2 \times \mathcal{L}^d_{rec}(\boldsymbol{x})$
  - $\phi \leftarrow \phi - \alpha \nabla_\phi \sum_{\boldsymbol{x} \in X^m} \left(\mathcal{L}_{rec}(\boldsymbol{x}) + \beta \mathcal{L}_{KL}(\boldsymbol{x}) \right)$

return θ, ϕ

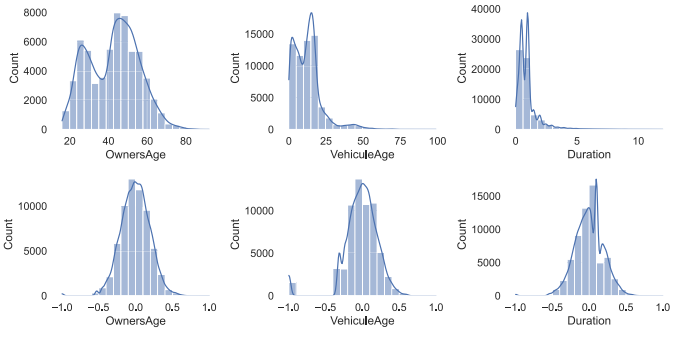


Fig. 6. Distributions of *OwnersAge*, *VehicleAge*, and *Duration* before (upper plots) and after (lower plots) applying a normal quantile transformation and a min–max scaling to the range $(-1, 1)$.

the model (i.e., reconstruction accuracy, latent space properties, and synthetic data quality) and its ability to learn a meaningful latent

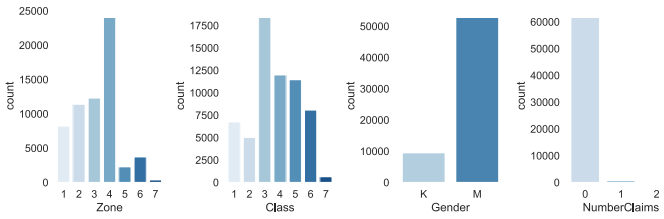


Fig. 7. Imbalanced classes in variables *Zone*, *Class*, *Gender* and *NumberClaims*.

space. If certain classes are underrepresented in the training subsets, whose size are determined by the batch size hyper-parameter, 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. To ensure each training subset includes enough representative examples from the minority classes, we recommend the use of smaller batch sizes 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 subsections, we find that choosing a moderate batch size of 256 ensures minority classes representation while still remaining within acceptable computation time (around 20 min). The ability of our VAE to handle imbalanced classes, that is, its ability to reconstruct and generate new policies representing all classes, including those with fewer observations, is illustrated in the following subsections in Figs. 12 and 17. The following Section 5.2 further details the hyper-parameters set up and model architecture that lead to the model performance.

5.2. Model architecture and hyper-parameters 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 architecture of our encoder and decoder networks, as well as the values of the key hyper-parameters involved in the training of our VAE are listed in Tables 3 and 4. Experimentation and tuning are often necessary to find optimal values for these hyper-parameters depending on the input data and its complexity, and the desired properties of the learned latent space. 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 SELU activation function. By definition, an activation function determines whether a neuron should be activated by combining the sum of the weighted inputs and a bias term. SELUs are activation functions that induce self-normalisation (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, & 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 initialiser. The variational parameters are initialised with zeros (for the means), and a small constant value set to 0.1 (for the log variances).

The latent (hidden) representation space is typically referred to as a bottleneck layer (Tishby, Pereira, & 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

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

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 3
Structure and configuration of the encoder and decoder hidden layer(s) and nonlinear activation functions (see Fig. 19).

	Encoder network	Bottleneck layer	Decoder network
# of layer(s)	1	1	2
# of neurons per layer	22	15	(22, 22)
Activation function	SELU	–	(SELU, (tanh, softmax))
Initialiser(s)	Lecun normal	$\mu = 0, \sigma = 0.1$	(Lecun normal, Xavier)

Table 4
Choice of the optimiser, its learning rate, and the gradient descent hyper-parameters.

Hyper-parameter	Value
RMSprop learning rate	0.001
Batch size	256
Number of epochs	75

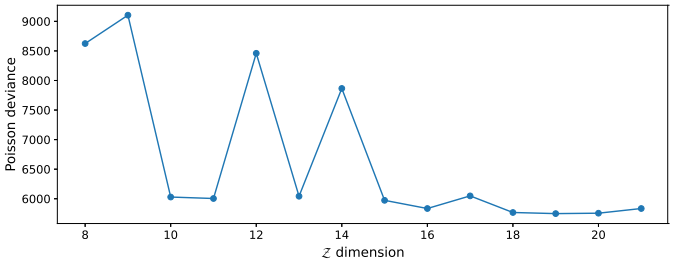


Fig. 8. Evolution of the Poisson deviance of a GLM model trained on the reconstructed insurance portfolio to predict the claim frequencies of the authentic policies, with respect to the latent space $\mathcal{Z}$ dimension $d \in (8, 21)$.

owner's risk profile. The choice of setting the latent space dimension d to 15, thereby reducing the dimension of about 1/3, is based on an extensive analysis of the quality of both the reconstructed and synthetic data. The metrics and visualisations used to choose the latent space dimension are detailed in the following two subsections for the optimal value of $d = 15$. Fig. 8 gives some insights into this choice by showing the evolution of the Poisson deviance with respect to the latent space dimension d. This deviance metric evaluates the quality of the reconstructed portfolio by estimating the goodness of fit of a GLM model trained on the reconstructed insurance portfolio to predict the claim frequencies of the authentic policies. To train our regression model (GLM) on the reconstructed portfolio, we extracted the covariates `Zone`, `Class`, `Gender`, `OwnersAge`, and `VehicleAge` from the decoder's output. We first discretised the continuous variables `OwnersAge` and `VehicleAge` before one-hot-encoding all covariates. We regressed the target variable `NumberClaims` (extracted from the decoder's output) given the one-hot-encoded covariates with a sample weight equal to the `Duration` (also extracted from the decoder's output). We can then use this trained GLM as an estimator to predict the claim frequencies $\hat{\lambda}_i$ of the original portfolio and compute the Poisson deviance, defined as:

$$D_{Poisson} = 2 \sum_{i=1}^{n} N_i \left(\frac{v_i}{N_i} \hat{\lambda}_i - \left(\log \frac{\hat{\lambda}_i v_i}{N_i} + 1 \right) 1_{\{N_i \geq 1\}} \right). \tag{13}$$

where N_i/v_i are the observed claim frequencies of the authentic policies (obtained by dividing *NumberClaims* by *Duration* in the original dataset) and $\hat{\lambda}_i$ are the predicted frequencies of the original policies estimated by the GLM trained on the reconstructed policies. The closer this deviance is to the deviance of a GLM trained on the original portfolio, i.e., 5961.84, the better our reconstruction quality is. To give some insights into the impact of the latent space $\mathcal{Z}$ dimension d on the reconstruction quality (measured by the Poisson deviance in Fig. 8), we provide a brief analysis of its impact on the quality of the reconstructed categorical variables. With $d = 1$, only the most represented modality in each categorical variable (see Fig. 7) is reconstructed. When we increase the $\mathcal{Z}$ dimension to two, our model is able to reconstruct additional categories (Zone 2, Class 1, 3, 5, and 7). With $d = 3$, while almost all modalities of the variable Class are reconstructed (except Class 2, with the second lowest number of observations), only Zone 4, Male, and 0 reported claims are reconstructed. With $d = 4$, we start to see real improvements in the categorical variables reconstruction (all modalities of the variable Class, Gender, and Zone (except Zone 7 with the lowest number of observations), are reconstructed). As we increase the $\mathcal{Z}$ dimension to five, six, and seven, we improve the misclassification error, yet, we still fail to reconstruct Class 2, with the second lowest number of observations. As of $d = 8$, the model is able to reconstruct all modalities. However, the reconstructed portfolio still deviates too much from the authentic as the GLM trained on the reconstructed fails to accurately predict the claim frequencies of the authentic portfolio as illustrated in Fig. 8 (increase in deviance of

Table 5
Latent means and standard deviations.

	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

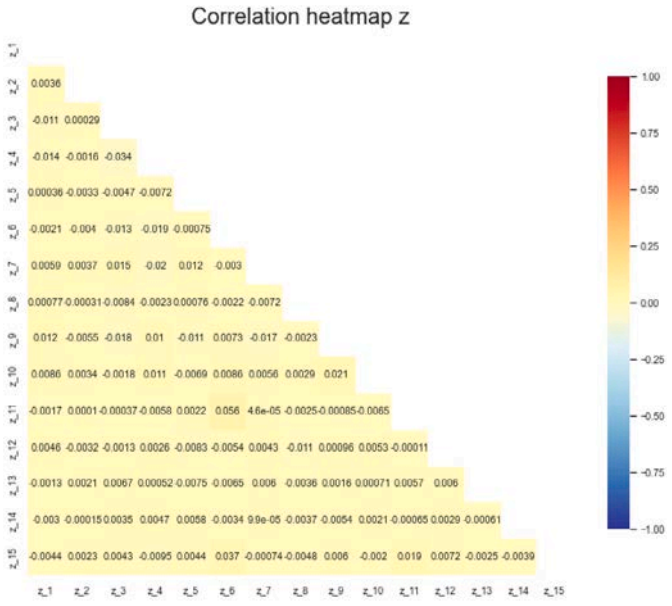


Fig. 9. Pearson's correlation between the latent codes $(Z_j)_{j=1,\ldots,d}$.

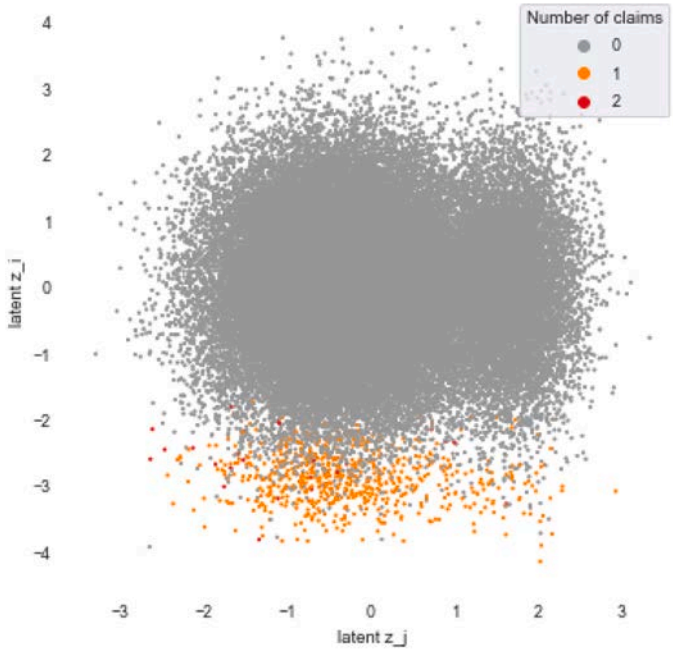


Fig. 10. Scatter plot between the observed values of two latent variables Z_i and Z_j.

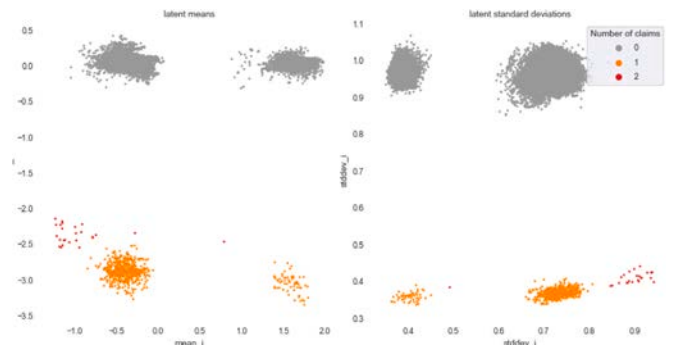


Fig. 11. Scatter plot of the latent means and standard deviations between the observed values of two latent variables Z_i and Z_j.

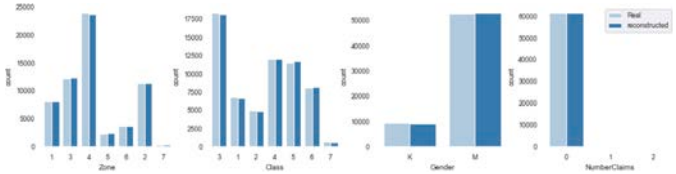


Fig. 12. Comparison of real (light blue) and reconstructed (dark blue) categorical data distributions (frequency of each observed category). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

51.55%). We first observe a deviance drop, with a deviance closer to the reference value of 5961.84, with $d = 10$, 11, and 13. Starting from $d = 15$ (chosen as the optimal value), the deviance stabilises.

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 initialiser. Because of the heterogeneity in the distributions of our input data (see Section 4.3), the decoding layer outputs are sliced before being passed through tanh and softmax activation functions with Xavier initialisers in the final layer of our decoder (as illustrated in Figs. 5 and 19).

The RMSprop optimiser with default learning rate value $\alpha = 0.001$ (which determines the speed of the optimiser's descent on the error curve) gave better results on our training data than the often-recommended Adam optimiser. The main hyper-parameters of the (mini-batch) gradient descent optimisation 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 shuffled dataset are used 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 regularising effects and improve the model's generalisation — 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 generalisation. 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 (around 20 min).

Thanks to the KL divergence regulariser term in the loss function, our VAE was able to learn multivariate independent latent variables. As shown in Fig. 9, the linear correlations between the latent encodings are

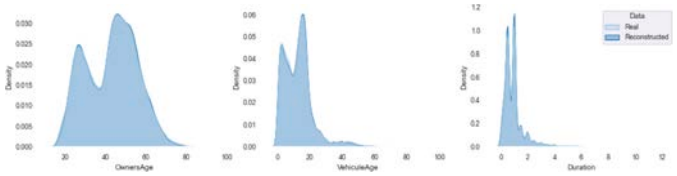


Fig. 13. Comparison of real (light blue) and reconstructed (dark blue) continuous data distributions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

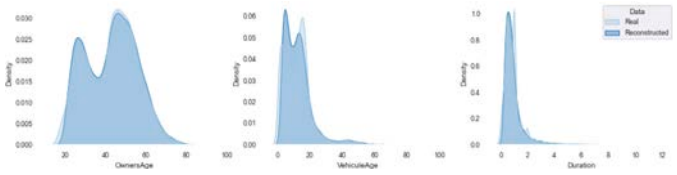


Fig. 14. Reconstructed (in dark blue) *Duration*, *OwnersAge* and *VehicleAge* without quantile transformation compared to their real distributions (in light blue). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

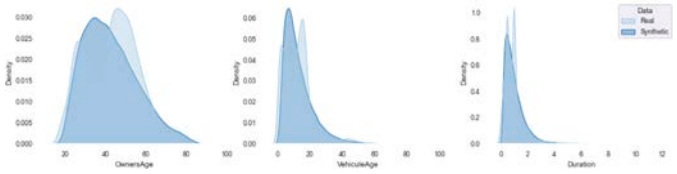
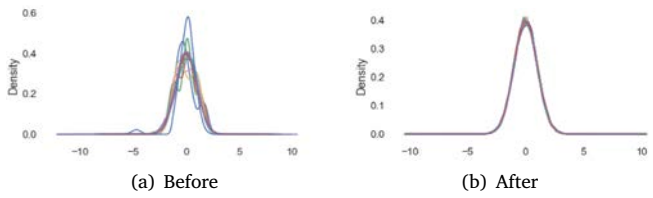


Fig. 15. Synthetic (in dark blue) *Duration*, *OwnersAge* and *VehicleAge* without quantile transformation compared to their real distributions (in light blue). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) Before (b) After

Fig. 16. Densities of the $d = 15$ latent variables, before and after normal quantile transformation.

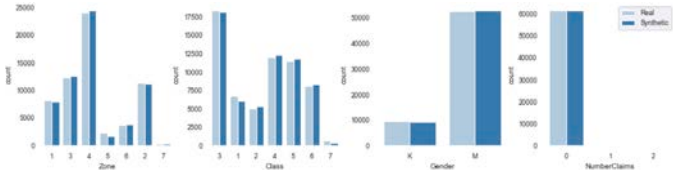


Fig. 17. Comparison of real (light blue) and synthetic (dark blue) categorical data distributions (frequency of each observed category). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

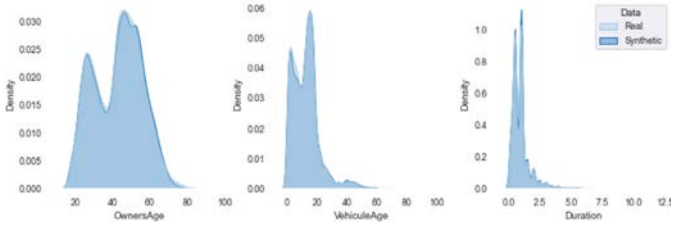


Fig. 18. Comparison of real (light blue) and synthetic (dark blue) continuous data distributions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

close to 0. The latent variables are also centred around 0 with standard deviations close to 1 (see Table 5 and Fig. 11). Thanks to the reconstruction loss, which allows for deviations from the unit Gaussian prior to capture the data most salient features, we also observe that policies which reported 1 or 2 motor claims are concentrated towards a lower region of the latent space (see Fig. 10) and that their mean and variance also slightly deviate from the standard values of 0 and 1 as illustrated in Fig. 11. This enabled our VAE model 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 $\hat{\boldsymbol{x}}$; 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. 12, the frequency of each observed category in the reconstructed data matches that of the original data. Fig. 13 also shows that the densities of the original and reconstructed continuous data are almost perfectly overlapping.

The visual comparison between Figs. 13 and 14 illustrates the positive impact of introducing and applying a quantile transformation on the continuous variables (following non-Gaussian multi-modal distributions), before inputting them into the encoder network. Without applying our normal quantile transformation on the inputted continuous variables, the distributions of the reconstructed variables (in dark blue) clearly fail to capture the two modes of the original *VehicleAge*'s distribution and the multiple modes of the original variable *Duration*. The impact of introducing a quantile transformation on the multi-modal input variables is even more pronounced when comparing the synthetic data distributions in Figs. 15 and 18 (obtained in Section 5.3 by feeding random noises to the decoder's network). When training our VAE on our pre-processed data without a normal quantile transformation scaler, the synthetic distributions (in dark blue) in Fig. 15 completely fail to reproduce the modes of the original distributions (in light blue). As illustrated in Fig. 18, the normal quantile transformation we proposed to scale our continuous input data greatly improves the distributions of the synthetic continuous data.

5.3. Synthetic data generation

As motivated in Section 3, the trained decoder network can 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 difficult to collect. 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 & Welling, 2013) nor (Rezende et al., 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 $p(\boldsymbol{z}) \sim \mathcal{N}(0, \boldsymbol{I})$ and the difference matters. As illustrated in the left plot of Fig. 16, although the latent variables are roughly centred around 0 with standard deviations close to 1 (see Table 5), 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. 16). With that knowledge in mind, we generate new policies by first sampling from

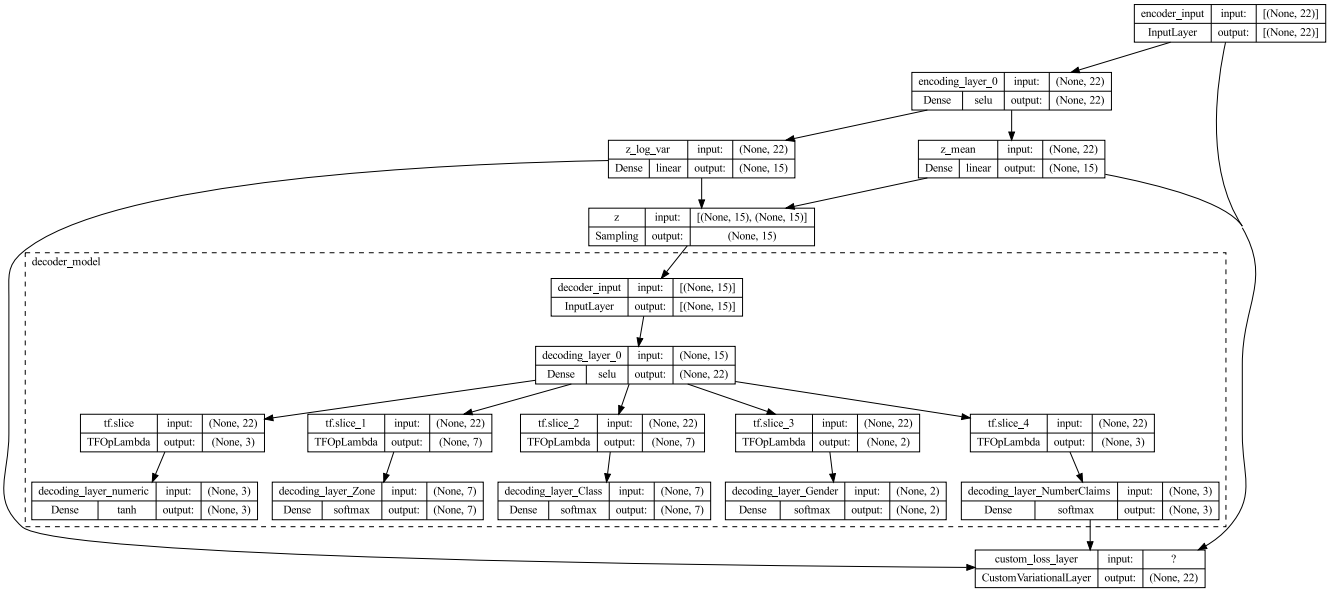


Fig. 19. Model architecture.

Table 6
Pearson and Spearman correlations between real and synthetic continuous data.

		OwnersAge		VehicleAge	
		Pearson ρ	*Spearman* r_s	*Pearson* ρ	*Spearman* r_s
VehicleAge	Real	0.0755	0.0671		
	Synthetic	0.0770	0.0777		
	Score	99.9243%	99.4729%		
Duration	Real	0.1173	0.1694	−0.0278	0.0787
	Synthetic	0.1334	0.1689	0.0526	0.0608
	Score	99.1948%	99.9766%	95.9783%	99.1068%

the prior distribution $p(\boldsymbol{z}) \sim \mathcal{N}(0, I)$. By doing so, samples $\boldsymbol{z}$ from the whole data distribution are drawn. Because of the non-bell shaped distributions of the latent codes $\boldsymbol{z}$, sampling from $\mathcal{N}(0, I)$ might fail and under-sample or omit some regions of the true distribution, over-sample others, and even provide inputs not covered by the aggregate posterior. Therefore, we propose to apply the inverse quantile transformations of the latent variables to the normally distributed samples $\boldsymbol{z}$.

The new samples $\boldsymbol{z}$ are then fed as input to the decoder. The quality of the so-generated synthetic policies is evaluated with respect to the statistical characteristics of the original portfolio. Figs. 17 and 18 show that the distributions of the synthetic data match those of the original data. The ability of our VAE model to generate new policies representing all classes, including those with fewer observations (e.g., Class 7 or Zone 7), was achieved thanks to the model architecture (including the heterogeneous reconstruction loss with a categorical cross-entropy for categorical variables) and the hyper-parameters setup (including the choice of a moderate batch size to include enough representative examples from the minority classes in the training subsets) detailed in the previous Section 5.2.

As mentioned at the end of the previous Section 5.2 and illustrated in Fig. 18, the normal quantile transformation we introduced and applied to scale our continuous variables (before inputting them into the encoder network) greatly improved the similarity between the synthetic continuous variables distributions and the original ones.

Table 7
Kolmogorov–Smirnov statistic-based metric between real and synthetic continuous data columns.

	OwnersAge	VehicleAge	Duration
KS metric	0.9779	0.9856	0.9863

This visual analysis of the quality of our synthetic data is further supported and verified by a wide range of measures. We first evaluate the ability of our VAE model to capture pairwise correlations between continuous variables by computing Pearson and Spearman correlations. Table 6 shows that the original correlations between continuous variables are preserved in the synthetic dataset. The scores in Table 6 were obtained by computing $1-0.5\left|\text{corr}^{real} - \text{corr}^{synthetic}\right|$. The synthetic and original continuous data distributions are further compared using the Kolmogorov–Smirnov (KS) statistic-based metric. The results are reported in Table 7. The output values are pretty close to 1, indicating that the distributions (the real CDFs and the synthetic CDFs values) are almost identical, verifying our visual analysis.

The boundaries adherence and statistical similarities reported in Table 8 are also verified. All synthetic data are within the minimum and maximum bounds of the real data, and we observe highly similar values for the mean, standard deviations, and different quantiles.

The quality of the synthetic categorical data is checked using the Chi-Squared test-based metric. The results reported in Table 9 are

Table 8
Statistic t similarity between real and synthetic continuous data and statistical score $1 - \frac{\|t^{real} - t^{synthetic}\|}{max^{real} - min^{synthetic}}$ for the mean, standard deviation, and median.

	t	OwnersAge		VehicleAge		Duration	
		Real	Synthetic	Real	Synthetic	Real	Synthetic
Statistical similarity	*mean*	42.5438	42.8047	12.5584	12.6940	1.0079	0.9938
	std	12.9549	12.9445	9.6846	9.6192	1.0621	0.9759
	min	16.0000	16.0000	0.0000	0.0000	0.0027	0.0027
	25%	31.0000	31.0000	5.0000	6.0000	0.4904	0.4959
	50%	44.0000	44.0000	12.0000	12.0000	0.8877	0.9178
	75%	52.0000	53.0000	16.0000	17.0000	1.0000	1.0000
	max	92.0000	83.0000	99.0000	87.0000	11.9945	11.7653
Statistical score	*mean*	0.9966		0.9986		0.9988	
	std	0.9999		0.9993		0.9928	
	median	1.0000		1.0000		0.9975	

Table 9
Chi-squared test, and classes coverage between real and synthetic categorical variables.

	Zone	Class	Gender	NumberClaims
Chi-Squared test	0.9852	0.9815	0.9954	0.9977
Modalities coverage	100.0000%	100.0000%	100.0000%	67.6667%

Table 10
Contingency score between real and synthetic categorical variables.

	Zone	Class	Gender
Class	0.9581%		
Gender	0.9796%	0.9738%	
NumberClaims	0.9843%	0.9797%	0.9931%

Table 11
Poisson deviances.

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

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).

We further compare the contingency similarity of the categorical variables in Table 10. These scores were obtained by means of a contingency table, which displays frequencies for combinations of two categorical columns.

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 in Eq. (13) obtained with two different GLMs.

The first GLM is trained on the original insurance portfolio X and is used to predict the claim frequencies $\hat{\lambda}_i^{GLM_1}$ of the original policies $i = 1, \dots, n$. By replacing in Eq. (13) the observed claim frequencies N_i/v_i by the number of claims in the original variable *NumberClaims* divided by the policy's exposure in the original variable *Duration*, and $\hat{\lambda}_i$ by the predicted claim frequencies $\hat{\lambda}_i^{GLM_1}$, we obtain a Poisson deviance equal to 5 691.01.

The second GLM is trained on the synthetic portfolio X^s and is used to predict the claim frequencies $\hat{\lambda}_i^{GLM_2}$ of the original policies $i = 1, \dots, n$. We were able to fit a GLM to the synthetic data because both the continuous variable *Duration* and the Multinomial variable *NumberClaims* were fed as input to our VAE. By replacing in Eq. (13) the observed claim frequencies N_i/v_i by the synthetic variable *NumberClaims* divided by the synthetic variable *Duration* (extracted from the generative decoder's output), and $\hat{\lambda}_i$ by the predicted claim frequencies $\hat{\lambda}_i^{GLM_2}$ of the original policies, we obtain a Poisson deviance equal to 5 961.84. Table 11 reports the Poisson deviances and the difference (in %) from the reference Poisson deviance obtained with the first GLM. The smaller this variation (%), the more our synthetic insurance portfolio can be considered a strong substitute for the real one. Fig. 20 illustrates the aforementioned results; subject to the same estimator (GLM), predicted claim frequencies from the authentic and synthetic portfolios should be close to the first bisector.

We further assess the quality of the reconstructed portfolio by fitting a third GLM on the reconstructed portfolio X^r and by predicting the claim frequencies $\hat{\lambda}_i^{GLM_3}$ of the original policies $i = 1, \dots, n$. By replacing in Eq. (13) the observed claim frequencies N_i/v_i by the reconstructed variable *NumberClaims* divided by the reconstructed variable *Duration* (extracted from the decoder's output), and $\hat{\lambda}_i$ by the predicted claim frequencies $\hat{\lambda}_i^{GLM_3}$ of the original policies, we obtain a Poisson deviance 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.

We finally assess to which extent policyholders could be personally identified in the synthetic data. To do so, we ensure the generated synthetic policies are 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 duplicates and 97.45% of the policies are unique. Among the unique fake policies, 1.82% are copies (and thence personally identifiable information) 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 impossible to distinguish them from the fake ones — without access to the real portfolio.

6. Conclusions

This article explored the application of VAEs for multi-type insurance data. The study's objective was to leverage the power of VAEs to extract a meaningful and low-dimensional representation of an insurance portfolio to generate artificial data. Thanks to our quantile transformations and NNs architecture, we successfully generated 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 while overcoming privacy concerns associated with the use of authentic policyholder information. Although VAEs assure a bright

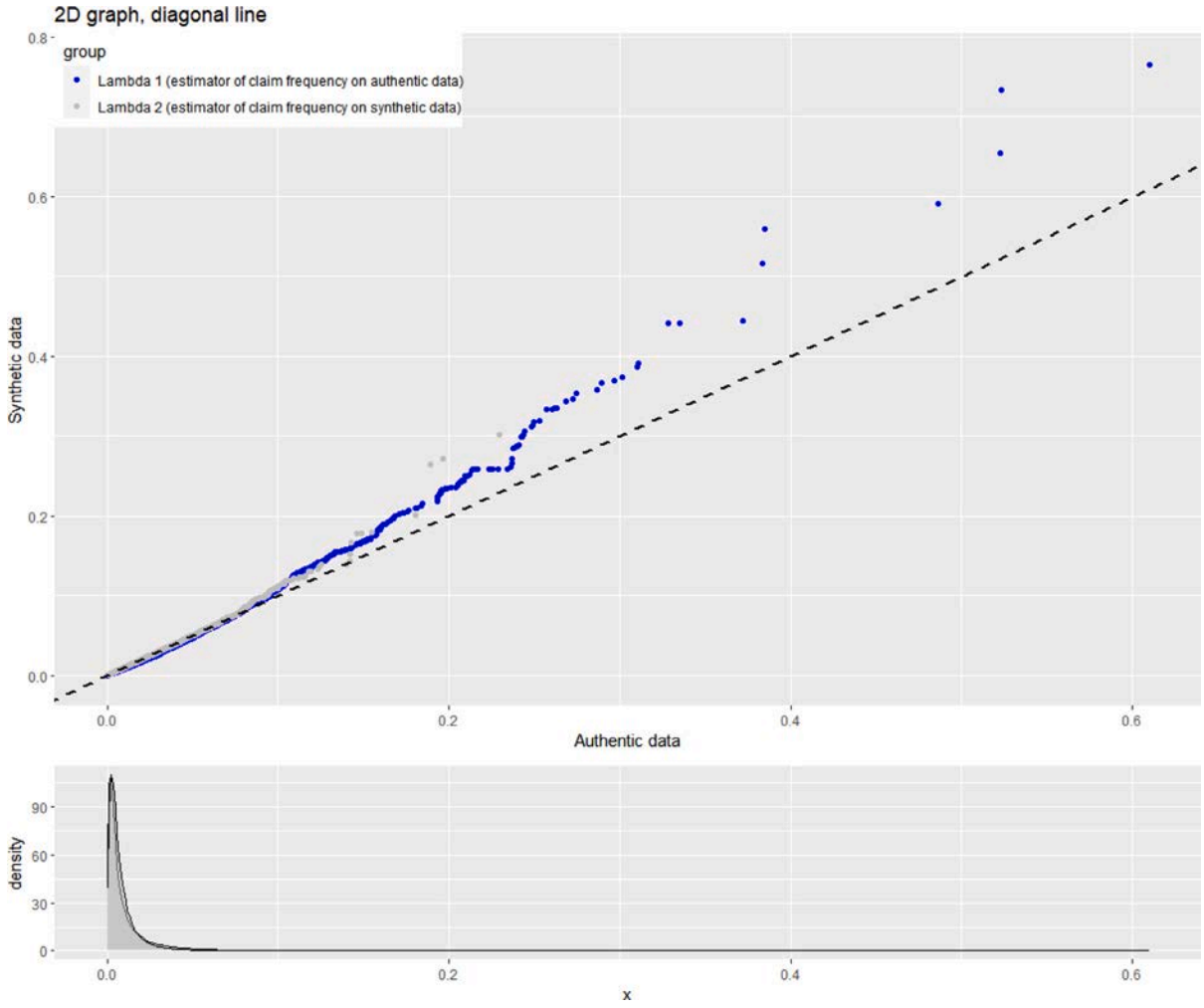


Fig. 20. Proximity to the 1st bisector of the synthetic and authentic predicted claim frequencies, where λ_1 and λ_2 denote the average claim numbers estimators respectively calibrated on the authentic $x_i^{authentic}$ and synthetic $x_k^{synthetic}$ data (where i and k denote the ith and kth policyholder in the authentic and synthetic portfolio). The blue dots refer to the pair of points $\left(\lambda_1\left(x^{authentic}\right), \lambda_1\left(x^{synthetic}\right)\right)$ and the grey dots refer to the pair of points $\left(\lambda_2\left(x^{authentic}\right), \lambda_2\left(x^{synthetic}\right)\right)$.

outlook for insurance analytics, there are still challenges to overcome. In particular, the research highlighted the significance of appropriate model architecture and hyper-parameters tuning to achieve optimal results. Careful consideration should be given to the selection of activation functions, layer sizes, regularisation techniques, and optimisation 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 utilising the blurred latent representation of the original rating factors in a downstream task, the direct and explicit usage of sensitive attributes, such as gender or ethnicity, in the subsequent regression task could be avoided. Finally, our generative model could also be adapted and conditioned on a modality or a target value to generate new additional representative policies.

CRediT authorship contribution statement

Charlotte Jamotton: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft, Writing – review & editing, Visualization. **Donatien Hainaut:** Conceptualization, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Funding

This work was supported by the project ASTeRISK under research grant ID 40007517, from the Excellence of Science (EoS) program call by FWO and FNRS.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Charlotte Jamotton reports financial support was provided by The Excellence Of Science (EOS). If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank the reviewers for their valuable suggestions and constructive comments, which significantly improved the quality of the manuscript.

Appendix

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

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

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

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

Proof (*Proposition 2*). By definition, the KL divergence is

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

where the densities are equal to

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

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

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

and

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

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

Input: Observed values $\{x_1, x_2, \dots, x_n\}$ of a continuous variable X whose empirical cumulative distribution function (CDF) is denoted $F(X)$.

Rank estimation: the CDF $F(X)$ 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 $q = \left\{q_1, q_2, \dots, q_{n_q}\right\}$ used to discretise $F(X)$. // They can be, e.g., evenly spaced: $q_i = \frac{i-1}{n_q-1} \times 100$ for $i = 1, 2, \dots, n_q$, and represent the desired percentiles of the target distribution G.

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

for *each* q_i **do**

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

$p_i \leftarrow$ Determine the quantile rank // i.e., determine the position of the q_i-th percentile within the sorted values $\boldsymbol{x}$ through interpolation where $\mathbb{Z}$ denotes the set of integer

$$p_i = \mathbb{I}_{\{k_i \in \mathbb{Z}\}}\left(x_{k_i} + \left(x_{k_i+1} - x_{k_i}\right)\left(k_i - \lfloor k_i \rfloor\right)\right) + \mathbb{I}_{\{k_i \notin \mathbb{Z}\}} x_{\lfloor k_i \rfloor}$$

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

$\boldsymbol{\Phi}^{-1}(p) \leftarrow$ Map to a normal distribution by applying the inverse of the desired distribution $\mathcal{N}(\mu,\sigma)$, denoted $\boldsymbol{\Phi}^{-1}$.

$X^{tr} \leftarrow$ Scale the transformed values $\boldsymbol{\Phi}^{-1}(p)$ to have a zero mean and unit variance

$$X^{tr} = \left(x_i\right)_{i=1,\dots,n}^{tr} = \frac{\boldsymbol{\Phi}^{-1}(p) - \mu_{\boldsymbol{\Phi}^{-1}(p)}}{\sigma_{\boldsymbol{\Phi}^{-1}(p)}}$$

Data availability

The dataset used in this study is available on the companion website of the book "Non-Life Insurance Pricing with Generalized Linear Models" by Ohlsson and Johansson (2010).

References

Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(4), 433–459.

Alazizi, A., Habrard, A., Jacquenet, F., He-Guelton, L., & Oblé, F. (2020). Dual sequential variational autoencoders for fraud detection. In *Advances in intelligent data analysis XVIII: 18th international symposium on intelligent data analysis, IDA 2020, konstanz, Germany, April 27–29, 2020, proceedings 18* (pp. 14–26). Springer.

An, J., & Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, *2*(1), 1–18.

Antelmi, L., Ayache, N., Robert, P., & Lorenzi, M. (2019). Sparse multi-channel variational autoencoder for the joint analysis of heterogeneous data. In *International conference on machine learning* (pp. 302–311). PMLR.

Baur, C., Wiestler, B., Albarqouni, S., & Navab, N. (2019). 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* (pp. 161–169). Springer.

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning: vol. 4*, Springer.

Bond-Taylor, S., Leach, A., Long, Y., & Willcocks, C. G. (2021). Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(11), 7327–7347.

Chakraborty, T., Ks, U. R., Naik, S. M., Panja, M., & Manvitha, B. (2024). Ten years of generative adversarial nets (GANs): a survey of the state-of-the-art. *Machine Learning: Science and Technology*, *5*(1), Article 011001.

Deasy, J., Simidjievski, N., & Liò, P. (2020). Constraining variational inference with geometric Jensen-Shannon divergence. *Advances in Neural Information Processing Systems*, *33*, 10647–10658.

Delong, Ł., & Kozak, A. (2021). The use of autoencoders for training neural networks with mixed categorical and numerical features. *ASTIN Bulletin: The Journal of the IAA*, 1–20.

Denuit, M., Hainaut, D., & Trufin, J. (2019). Effective statistical learning methods for actuaries III: Neural networks and extensions. *Springer actuarial*, Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-25827-6, URL: http://link.springer.com/10.1007/978-3-030-25827-6.

Duan, Y., Wang, L., Zhang, Q., & Li, J. (2022). Factorvae: A probabilistic dynamic factor model based on variational autoencoder for predicting cross-sectional stock returns. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4468–4476).

Ferrario, A., Noll, A., & Wuthrich, M. V. (2020). Insights from inside neural networks. Available at SSRN 3226852.

García-Ordás, M. T., Benítez-Andrades, J. A., García-Rodríguez, I., Benavides, C., & Alaiz-Moretón, H. (2020). Detecting respiratory pathologies using convolutional neural networks and variational autoencoders for unbalancing data. *Sensors*, *20*(4), 1214.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 315–323). JMLR Workshop and Conference Proceedings.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. *Statistics*, *1050*, 10.

Guo, C., & Berkhahn, F. (2016). Entity embeddings of categorical variables. arXiv preprint arXiv:1604.06737.

Hainaut, D. (2018). A neural-network analyzer for mortality forecast. *ASTIN Bulletin: The Journal of the IAA*, *48*(2), 481–508.

Hainaut, D. (2019). A self-organizing predictive map for non-life insurance. *European Actuarial Journal*, *9*(1), 173–207.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., et al. (2017). Beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*.

Huang, H., He, R., Sun, Z., Tan, T., et al. (2018). Introvae: Introspective variational autoencoders for photographic image synthesis. *Advances in Neural Information Processing Systems*, *31*.

Jang, E., Gu, S., & Poole, B. (2016). Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. *Advances in Neural Information Processing Systems*, *30*.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, *37*(2), 233–243. http://dx.doi.org/10.1002/aic.690370209, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Kusner, M. J., Paige, B., & Hernández-Lobato, J. M. (2017). Grammar variational autoencoder. In *International conference on machine learning* (pp. 1945–1954). PMLR.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K.-R. (2002). Efficient backprop. In *Neural networks: tricks of the trade* (pp. 9–50). Springer.

Li, X., & She, J. (2017). Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 305–314).

Li, B., Sun, Z., & Guo, Y. (2019). Supervae: Superpixelwise variational autoencoder for salient object detection. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 8569–8576).

Lim, J., Ryu, S., Kim, J. W., & Kim, W. Y. (2018). Molecular generative model based on conditional variational autoencoder for de novo molecular design. *Journal of Cheminformatics*, *10*, 1–9.

Lu, G., Zhao, X., Yin, J., Yang, W., & Li, B. (2020). Multi-task learning using variational auto-encoder for sentiment classification. *Pattern Recognition Letters*, *132*, 115–122.

Ma, C., Tschiatschek, S., Turner, R., Hernández-Lobato, J. M., & Zhang, C. (2020). VAEM: a deep generative model for heterogeneous mixed type data. *Advances in Neural Information Processing Systems*, *33*, 11237–11247.

Mancisidor, R. A., Kampffmeyer, M., Aas, K., & Jenssen, R. (2021). Learning latent representations of bank customers with the variational autoencoder. *Expert Systems with Applications*, *164*, Article 114020.

Mansour, R. F., Escorcia-Gutierrez, J., Gamarra, M., Gupta, D., Castillo, O., & Kumar, S. (2021). Unsupervised deep learning based variational autoencoder model for COVID-19 diagnosis and classification. *Pattern Recognition Letters*, *151*, 267–274.

Mescheder, L., Nowozin, S., & Geiger, A. (2017). Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *International conference on machine learning* (pp. 2391–2400). PMLR.

Miotto, R., Li, L., Kidd, B. A., & Dudley, J. T. (2016). Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, *6*(1), 1–10.

Nazabal, A., Olmos, P. M., Ghahramani, Z., & Valera, I. (2020). Handling incomplete heterogeneous data using vaes. *Pattern Recognition*, *107*, Article 107501.

Noll, A., Salzmann, R., & Wuthrich, M. V. (2020). Case study: French motor third-party liability claims. Available at SSRN 3164764.

Ohlsson, E., & Johansson, B. (2010). *vol. 174, Non-life insurance pricing with generalized linear models*. Springer.

Öğretir, M., Ramchandran, S., Papatheodorou, D., & Lähdesmäki, H. (2022). A variational autoencoder for heterogeneous temporal and longitudinal data. In *In 2022 21st IEEE international conference on machine learning and applications* (pp. 1522–1529). IEEE.

Pagnoni, A., Liu, K., & Li, S. (2018). Conditional variational autoencoder for neural machine translation. arXiv preprint arXiv:1812.04405.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning* (pp. 1310–1318). Pmlr.

Rezende, D., & Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning* (pp. 1530–1538). PMLR.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning* (pp. 1278–1286). PMLR.

Richman, R. (2021). AI in actuarial science–a review of recent advances–part 1. *Annals of Actuarial Science*, *15*(2), 207–229.

Semeniuta, S., Severyn, A., & Barth, E. (2017). A hybrid convolutional variational autoencoder for text generation. arXiv preprint arXiv:1702.02390.

Shen, X., Liu, B., Zhou, Y., Zhao, J., & Liu, M. (2020). Remote sensing image captioning via variational autoencoder and reinforcement learning. *Knowledge-Based Systems*, *203*, Article 105920.

Simard, P. Y., Steinkraus, D., Platt, J. C., et al. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*. Edinburgh.

Song, T., Zhang, X., Ding, M., Rodriguez-Paton, A., Wang, S., & Wang, G. (2022). DeepFusion: A deep learning based multi-scale feature fusion method for predicting drug-target interactions. *Methods*, *204*, 269–277.

Suh, S., & Choi, S. (2016). Gaussian copula variational autoencoders for mixed data. arXiv preprint arXiv:1604.04960.

Tingfei, H., Guangquan, C., & Kuihua, H. (2020). Using variational auto encoding in credit card fraud detection. *IEEE Access*, *8*, 149841–149853.

Tishby, N., Pereira, F. C., & Bialek, W. (2000). The information bottleneck method. arXiv preprint physics/0004057.

Van Den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. *Advances in Neural Information Processing Systems*, *30*.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11).

Wang, W., Gan, Z., Xu, H., Zhang, R., Wang, G., Shen, D., et al. (2019). Topic-guided variational autoencoders for text generation. arXiv preprint arXiv:1903.07137.

Wei, J., & Zou, K. (2019). Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196.

Wu, C., Wu, F., Wu, S., Yuan, Z., Liu, J., & Huang, Y. (2019). Semi-supervised dimensional sentiment analysis with variational autoencoder. *Knowledge-Based Systems*, *165*, 30–39.

Wu, Z.-b., & Yu, J.-q. (2019). Vector quantization: a review. *Frontiers of Information Technology & Electronic Engineering*, *20*(4), 507–524.

Xie, Q., Dai, Z., Hovy, E., Luong, T., & Le, Q. (2020). Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, *33*, 6256–6268.

Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). Modeling tabular data using conditional GAN. URL: http://arxiv.org/abs/1907.00503, arXiv:1907.00503 [cs, stat].

Xu, W., Sun, H., Deng, C., & Tan, Y. (2017). Variational autoencoder for semi-supervised text classification. In *Proceedings of the AAAI conference on artificial intelligence*.

Xu, L., & Veeramachaneni, K. (2018). Synthesizing tabular data using generative adversarial networks. http://dx.doi.org/10.48550/arXiv.1811.11264, URL: http://arxiv.org/abs/1811.11264, arXiv:1811.11264 [cs, stat].

Zhang, C., Liang, S., Lyu, F., & Fang, L. (2020). Stock-index tracking optimization using auto-encoders. *Frontiers in Physics*, *8*, 388.

Zhang, X., Yang, Y., Yuan, S., Shen, D., & Carin, L. (2019). Syntax-infused variational autoencoder for text generation. arXiv preprint arXiv:1906.02181.

Zhao, S., Song, J., & Ermon, S. (2017). Towards deeper understanding of variational autoencoding models. arXiv preprint arXiv:1702.08658.