Modeling Tabular Real Estate Transaction Data with Deep Generative Models

Master Thesis

Benjamin de Jong

04/02/2022

Modeling Tabular Real Estate Transaction Data with Deep Generative Models

Analysing how well Deep Generative Models can learn the underlying joint probability distribution of heterogeneous tabular real estate transaction data

Author: Benjamin de Jong

Submitted to:

Vrije Universiteit Amsterdam
Faculty of Science
Business Analytics
De Boelelaan 1105
1081HV Amsterdam

Host organisation:

Clappform BV
Data Science Team
Parlevinker 17
1186ZA Amstelveen

Graduation Supervisor:
Dr. Rikkert Hindriks

External Supervisor:
Ir. Jeroen Schoonderbeek

Second Reader:
Dr. Jakub Tomczak

04/02/2022

Preface

This thesis is written in order to obtain my MSc. degree and to finalize the two-year master's program Business Analytics at the Vrije Universiteit Amsterdam. Business Analytics is a multidisciplinary program, combining business mathematics and computer science. This research is conducted during a six-month internship at Clappform. Clappform is a software company that provides a cloud-based platform that enables advanced analytics using artificial intelligence and machine learning.

Commonly encountered problems during the development of artificial intelligence and machine learning models are scarce data availability, imbalanced features, and restrictions on privacy-sensitive data. This research investigates how well synthetic data can be used to overcome these obstacles. More specifically, this research analyzes how well deep generative models can learn the underlying joint probability distribution of heterogeneous tabular data in order to generate synthetic tabular data that could be used to overcome the mentioned obstacles.

Without Clappform, this research could not have been conducted. Therefore, I want to thank Clappform for giving me the opportunity and resources to fulfill this research. Thereby, I would like to thank the data analytics, product development, and business development team of Clappform, and in particular my external supervisor, Jeroen Schoonderbeek, for his continuous supervision and help throughout the internship. Finally, I want to thank my graduation supervisor of the Vrije Universiteit Amsterdam, Rikkert Hindriks, for his guidance and valuable input.

Management Summary

Nowadays, tabular data is the most encountered form of data in business. However, with the rise of the GDPR, available data cannot always be shared or used for analytical purposes. Thereby, when data is not subjective to the regulation of the GDPR, it often comes in scarce quantities or may be of low quality due to highly imbalanced categorical features. These restrictions limit the potential impact of data and make it challenging for data scientists to perform analyses and develop artificial intelligence products. However, synthetic data might be a solution to overcome such bottlenecks.

Synthetic data is artificially manufactured by an algorithm, such as a deep generative model, trained on empirical data. This implies that the synthetic data did not arise from real-world events whereas the empirical data did. However, high-quality synthetic data resembles (statistical) patterns and the analytical potential of the empirical data. Also, artificial intelligence and machine learning models trained on high-quality synthetic data should exhibit performance similar to the models trained on empirical data.

The company at which this research is conducted, Clappform, provides a flexible cloud-based platform that enables advanced analytics by utilizing artificial intelligence and machine learning. Clappform has noted that high-quality synthetic data could be of great value to acquire up to now unavailable data and to enhance the quality of their products. They were particularly interested in what impact synthetic data could have on the performance of their real estate rental price prediction model (ERV model), which is trained on (heterogeneous) tabular real estate transaction data. Therefore, this research answers the question:

How well can deep generative models learn real estate transaction data?

This concerns an analysis that concludes how well two deep generative models, a tabular variational autoencoder (TVAE) and a conditional tabular generative adversarial network (CTGAN), can learn the underlying probability distribution of heterogeneous tabular real estate transaction data.

The analysis can be divided into a statistical analysis and a machine learning efficacy analysis. The statistical analysis investigates how well deep generative models are able to reproduce statistical properties from the empirical real estate transaction data into synthetic data in terms of a similarity score. The machine learning efficacy analysis researches how well synthetic data can be used to replace empirical train data for the ERV model in terms of R^2. Thereby, it investigates what happens to the R^2 performance of the ERV model after augmenting empirical data with synthetic data, to balance imbalanced categorical features. Together, these analyses give a clear picture of how well deep generative models were able to learn the underlying distribution function of the real estate transaction data.

The results of the statistical analysis show that the TVAE model outperformed the CTGAN model with a similarity score of 0.88 against 0.84, respectively. Likewise, the ERV model performed best after being trained on synthetic data generated by the TVAE model with $R^2 = 0.7219$. This is a decrease of 14.78% with respect to the ERV model trained on empirical data. The best ERV model performance after being trained on synthetic data generated by the CTGAN was $R^2 = 0.3645$, which is a decrease of 43.15%. Based on the results of both analyses, it can be concluded for both models that there is a positive relationship between the train sample size and the quality of the synthetic data. Finally, augmenting the empirical train data with synthetic records to balance the imbalanced category "shell" for the categorical feature "interior type", resulted in no improvement of the ERV model on the total test set. However, the ERV model performance did increase with 9.28% on the shell test set, indicating that the ERV model became better at predicting rental prices for properties for which the interior type is shell.

How well a deep generative model can learn the underlying distribution of data strongly depends on its hyperparameter values. However, since the parameter search space for deep learning models can become arbitrarily large, a search for optimal hyperparameters is often infeasible within considerable time. Recent literature has shown that variable-length genetic algorithms can be used to improve hyperparameter settings within reasonable time and improve the performance of deep learning models. Therefore, future research is recommended to focus on how the performance of deep generative models on heterogeneous tabular data can be improved by optimization of hyperparameters with use of genetic algorithms.

Contents

Introduction

In recent years, there has been increasing interest in the field of Artificial Intelligence (AI). Machine Learning (ML), a subdomain of AI, includes creating algorithms that enable a computer to learn. Learning can be described as finding statistical properties and patterns in data by applying complex mathematical calculations (Nasteski, 2017). Using ML algorithms, processes, and workflows in organizations usually carried out by humans can intelligently be supported (Xu & Veeramachaneni, 2018).

This research is conducted in collaboration with Clappform. This is a software company that offers a flexible cloud-based platform that enables advanced analytics by using AI and ML solutions. Clappform's primary focus is on integrating such solutions into the business processes and workflows of its clients. However, impediments arise in sub-processes like data sharing, obtaining sufficient train and test data for model development and training supervised learning models on imbalanced data.

First, privacy is an important aspect, and proper privacy-preserving methods are required to share and process data (Puri & Haritha, 2016). Techniques that have been used to preserve privacy are identifier removal, quasi-identifiers usage, and value perturbation. However, such approaches suffer from limitations such as linkage attacks. A linkage attack is an attempt to re-discover personal information in anonymized data. For example, by combining quasi-identifiers, such as postal code, gender, and date of birth, that are present in the anonymized data and could be observed physically or found in publicly available data. Therefore, these approaches are not applicable (Park et al., 2018). Second, the amount of available train and test data for model development may be of insufficient size and data can only be used for testing if the company that produces this data is compliant with privacy regulations like the General Data Protection Regulation (GDPR). Although the availability constraint could be solved by manually gathering and labeling additional data, both might be difficult, time-consuming, and expensive (Nikolenko, 2021). Third, one of the fundamental problems in ML is imbalanced classes in data. A supervised learning algorithm tends to favor the majority class when learning from data in which the amount of training data for a minority class is substantially smaller than for the majority class (Nguyen et al., 2008).

Synthetic data could be a solution to these constraints. Synthetic data is artificially manufactured by an algorithm trained on empirical data. This implies that the synthetic data did not arise from real-world events whereas the empirical data did. High-quality synthetic data resembles (statistical) patterns and the analytical potential of the empirical data. Also, ML models trained on high-quality synthetic data should exhibit performance similar to the models trained on empirical data.

One of the fields in which synthetic data first appeared to be useful is computer vision. Computer vision algorithms, such as image classification and segmentation, require a large number of labeled train images to achieve an acceptable level of performance. If this level is not reached, the usefulness and potential impact of such a model are neglectable (Cho et al., 2015). Augmenting the train data by manually collecting and labeling images might become so laborious that it is considered not worth the effort (Nikolenko, 2021). Alternatively, generating labeled synthetic images with a pre-trained model to augment the train data is labor-free. Deep generative models such as Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) are appealing models for synthetic image generation (Xu & Veeramachaneni, 2018).

Nevertheless, it appears that tabular data is the most frequently encountered form of data in business (Xu & Veeramachaneni, 2018). Tabular data is structured data consisting of rows and columns. Whereas images only consist of continuous values, originating from a Gaussian-like distribution (Xu et al., 2019) (i.e., homogeneous data), tabular data might contain continuous, discrete, categorical, or a combination of these types (i.e., heterogeneous data). It is not self-evident that models like VAEs and GANs generate similar quality synthetic data for heterogeneous tabular data as for homogeneous (tabular) data. More specifically, (Ma et al., 2020) state that, due to heterogeneity of natural data sets, deep generative models often perform poorly in real-world applications.

Tabular data is used in applications and business functions such as marketing, research, and machine learning, in industries like finance, healthcare, and social media (Wagner, 2020). Nowadays, social media is a marketing and promotion opportunity that directly links companies to their clients (Durgam, 2018). Collecting (heterogeneous) tabular data from social media users such as, among other things, age, gender, education, and income level, allows to observe and research human behavioral patterns on an unprecedented scale (Zafarani et al., 2014). However, by regulation of the GDPR, data related to an identified or identifiable individual cannot be used for data processing. The same holds for personal data that could be used for business functions that include machine learning model development. Synthetic tabular data could be a solution to mitigate these limitations.

One of Clappform's products includes a rental price forecasting model (the ERV model) and uses, among other things, tabular real estate transaction data as train data. This data is heterogeneous since it contains mixed-type features following different marginal distributions. Despite the transaction data not containing any personal and privacy-sensitive information, synthetic real estate transaction data could still be valuable when for example the data owner wants to share the data with Clappform for analysis but does not want to share the original records. In addition, value could be created if the predictive performance of the ERV model exhibits similar performance when replacing the original train data with synthetic data. On top of that, value is created if the predictive performance of the ERV model improves when augmenting the original train data with synthetic data to balance imbalanced categorical features. All three cases motivate to investigate if synthetic data could be of value to Clappform.

The authors (Xu et al., 2019; Xu & Veeramachaneni, 2018) show that if a series of reversible data transformations are applied to the heterogeneous tabular data, it can be used as input for deep generative models to generate synthetic tabular data. Therefore, the contribution of this research is to provide insight into how well deep generative models can learn tabular real estate rental data.

More specifically, this research aims to examine how well synthetic tabular real estate transaction data, generated by deep generative models, preserves statistical properties from the empirical data, can be used to replace empirical train data for the ERV model, and can be used as augmentation data to balance imbalanced train data in order to improve the predictive performance of the ERV model.

This leads to the main research question:

- How well can deep generative models learn tabular real estate transaction data?

With sub-questions:

- How well can deep generative models reproduce statistical properties from the empirical real estate transaction data in synthetic real estate transaction data?
- How well performs the ERV model, trained on synthetic real estate transaction data, compared to the ERV model trained on empirical real estate transaction data?
- What is the effect of the train data size on how well deep generative models can learn real estate transaction data?
- What happens to the performance of the ERV model after balancing an imbalanced feature in the empirical real estate transaction data with synthetic records?

This thesis is organized as follows. First, chapter 4 introduces the reader to general concepts regarding machine learning, deep learning, and deep generative modeling. Hereafter, chapter 5 gives an overview of related work. Chapter 6 extensively describes the real estate transaction data with applied pre-processing steps. The models used to learn the real estate transaction data and methods to quantify how well the models learned are described in chapter 7. The experimental setup is described in 8, followed by the results of the experiment in chapter 9. Chapter 10 presents conclusions and the answer to the research question. Finally, the discussion and recommendations for further research are presented in chapter 11.

Preliminaries

Before getting deeper into the concept of how deep generative models can learn the probability distribution of heterogeneous tabular data, this section elaborates in an introductory manner on what synthetic data is and on general concepts related to machine learning, deep learning, and deep generative modeling.

4.1 Synthetic Data

In contrast to empirical data originating from real-life events, synthetic data is artificially created. This can be done in three ways. First, by using perturbation methods. Second, by combining or aggregating features, and third by sampling from the probability distribution from which empirical data originates (Brenninkmeijer & Hille, 2019). This research will focus on the latter case, in which deep generative models (DGMs) are used to learn the probability distribution of heterogeneous tabular real estate transaction data and sample synthetic data from that distribution.

4.2 Neural Network

An (artificial) neural network (NN) consists of three or more connected layers containing nodes. The first and the last layer are the input and output layers, respectively. Layers in-between the input and output layer are called hidden layers. Nodes in adjacent layers are connected through connections, called weights. The left part of figure 4.1 shows an example of a simple fully connected feed-forward neural network. Fully connected indicates that all nodes in successive layers are connected. Feed-forward implies that data enters the network via the input layer, gets propagated forward through the network, and exits the network via the output layer.

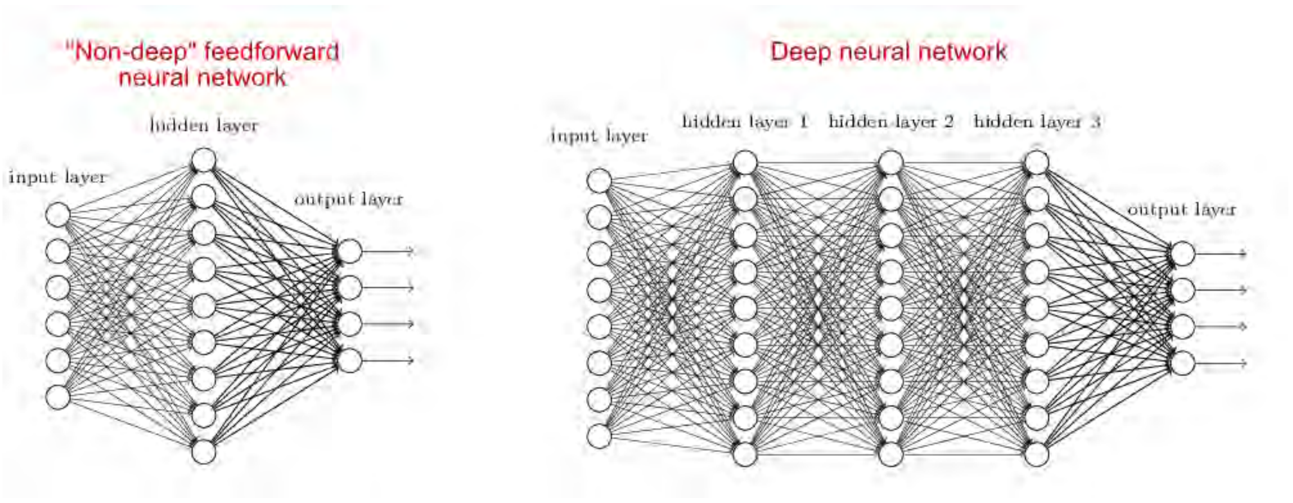


Figure 4.1: An example of a simple fully connected feed-forward neural network (left) and a fully connected deep neural network (right).

In a fully connected network, the total number of parameters (S) is represented by the sum of the number of weights (W) and biases (B) in the network. The total number of parameters is also referred to as the size of the network. Let the total number of layers be L and let V_l, W_l, and B_l be the number of nodes, weights, and biases in layer $l \in L$, respectively. Then, the size S_l for a single layer l is calculated as follows:

$$S_l = W_l + B_l \text{ , with} \tag{4.1}$$

$$W_l = V_{l-1} * V_l \text{ and } B_l = V_l \tag{4.2}$$

Hereafter, the size of the network can be calculated as:

$$S = \sum_{i=1}^{L} S_i \tag{4.3}$$

The sum goes over layer 1 to L and not over 0 to L, since the input layer has no preceding layers and should therefore not be included when calculating the size of a network.

The learning process of a neural network consists of two reiterative actions. First, data is passed forward through the network. This is called forward propagation and is discussed in subsection 4.2.1. Hereafter, a loss function, specified over the nodes in the output layer calculates the error made for this particular forward pass. Then, according to the size of the error, the parameters (i.e., weights and biases) in the network are adjusted. In contrast to forward propagation, the parameter adjustment is made in reversed order, starting at the output layer. Updating the parameters is called backpropagation and is discussed in subsection 4.2.2.

4.2.1 Forward propagation

Since the structure of the feed-forward computation is equivalent for each layer, let's consider the computation step between one pair of layers for convenience.

First, let x = $\{x_1, ..., x_{V_0}\}$ be (a sample of) data, used as input for the network. Second, let $w_{ji}^{(l)}$ be the weight going from node i in layer $l-1$ to node j in layer l and let $b_j^{(l)}$ be the bias connected to node j in layer l, $i \in \{1, ..., V_{l-1}\}$, $j \in \{1, ..., V_l\}$, $w_{ji}^{(l)}$, $b_j^{(l)} \in \mathbb{R}$. Third, let $a_i^{(l)}$ be the activation value for node i in layer l. For the input layer it holds that $a_i^0 = x_i$ Finally, let $f(\cdot) : \mathbb{R} \to \mathbb{R}$ be a non-linear activation function used to calculate the activation value for each node in layer l. Examples of non-linear activation functions are *sigmoid*, *tanh* and *ReLU*.

The activation value for each node in layer l is constructed in two steps. First, a linear combination is constructed of the form

$$h_j^{(l)} = \sum_{i=1}^{V_{l-1}} w_{ji}^{(l)} \, a_i^{(l-1)} + b_j^{(l)} \tag{4.4}$$

resulting in a "non-activation" value $h_j^{(l)}$. Second, the differentiable non-linear activation function $f(\cdot)$, transforms the non-activation value into the activation value

$$a_j^{(l)} = f(h_j^{(l)}) \tag{4.5}$$

This is done for each node j in layer l (Bishop, 2006). The activated layer l will then be used, together with the weights and biases of the next layer, to compute the activation values for the nodes in the next layer. When the activation values in the output later are calculated, the complete forward propagation step has finished.

To be computationally efficient, the computation between two layers is performed by matrix multiplications. The weights, including biases, non-activation values, and activation values are represented in matrices. After the multiplication of the weights, biases, and activation values from the preceding layer, the non-linear activation function is executed on all non-activation values in the resulting matrix, assuming that all layer nodes use the same non-linear activation function. This results in a matrix containing activated values.

4.2.2 Backpropagation and Gradient Descent

The size of a NN can become very large when increasing the number of layers and nodes per layer. Each network configuration (i.e., a combination of parameter values) might result in different behavior of the network. After forward propagating data through a network, its behavior is assessed by the loss function. After evaluating the loss of a forward propagation step, the derivatives of the loss function with respect to the parameters are calculated. Hereafter, these derivatives are used to compute the necessary adjustments to be made to the parameters (Bishop, 2006). Forward and backward propagation steps alternate until a sufficient configuration is found.

Most training algorithms aim to find a configuration in the parameter space for which the loss is as low as possible (Bishop, 2006). Formally, assuming that θ covers the parameter space, this can be written as:

$$arg\ min_{\theta}\ loss_{data}\ (\theta) \tag{4.6}$$

Gradient descent is a technique to calculate parameter adjustments. This algorithm evaluates for the current configuration, in which direction (i.e., gradient) of the loss landscape it has to move in order to obtain a network configuration with the steepest loss descent. The main objective is to find the parameter configuration to which a global minimum in the loss space belongs. However, in contrast to convex loss landscapes for which a local minimum is inherent to a global minimum, difficulties arise for non-convex parameter spaces due to multiple local minima in which a model can get stuck during training. Figure 4.2 shows a convex (left) and non-convex (right) three-dimensional loss landscape.

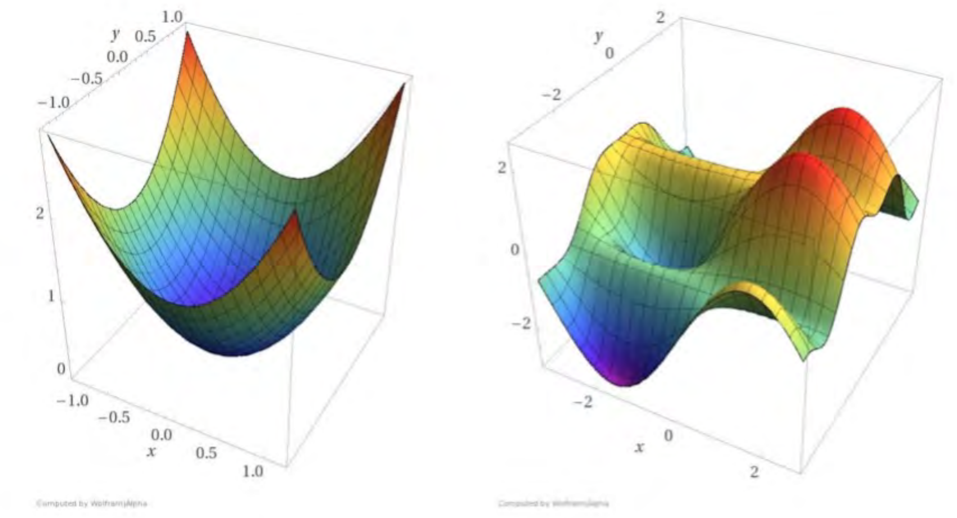


Figure 4.2: Three-dimensional loss landscapes for arbitrary parameters x,y and z. The loss landscape on the left is a convex parameter landscape whereas the loss landscape on the right is non-convex.

4.3 Deep Learning

A neural network with multiple hidden layers is called a deep neural network and belongs to the class of deep learning models (Abadi et al., 2016). The right part of figure 4.1 gives an example of a deep neural network, the feed-forward multi-layer perceptron. Other forms of deep neural networks are convolutional neural networks and recurrent neural networks. In contrast to the multilayer perceptron, the hidden layers of a convolutional neural network and a recurrent neural network consist of convolutional layers and recurrent layers, respectively.

For a deep neural network architecture, its model complexity defines to what extent the model can express complex functions. Factors that are of influence on model complexity are model framework, model size, optimization process, and data complexity (Hu et al., 2021). For example, this means that increasing the size of a network by augmenting the number of layers or the number of nodes per layer is inherent to increasing the complexity of a deep neural network.

Due to limited availability in computational power, the early architecture of neural networks did not contain multiple hidden layers. Since more significant amounts of computational power are available (especially GPU), more complex deep neural network architectures are feasible (Miikkulainen et al., 2019).

4.4 Deep Generative Models

Generative Models are unsupervised machine learning models, trying to learn regularities and patterns in data. Generative models such as Gaussian Mixture Models, Hidden Markov Models, and the Latent Dirichlet Allocation, assume that data originates from a probability distribution that can be approximated (Harshvardhan et al., 2020). This means that for tabular data, a generative model tries to learn the joint probability distribution over all present features in the data, considering the features being random variables.

Deep generative models (DGMs) are generative models based on deep neural network architectures and belong to the class of unsupervised deep learning techniques. These techniques have set a new standard for modeling complicated and high-dimensional probability distributions from a finite number of records and generating synthetic records from that distribution (Chen et al., 2018). This claim is substantiated by (Ruthotto & Haber, 2021), who make the exact same statement but specifically mention DGMs instead of unsupervised deep learning techniques in general.

Deep generative models can roughly be divided into Autoregressive, Flow-Based, and Deep Latent Variable Models (DLVMs). Among these models, distinction should be made between explicit and implicit models (Maaløe et al., 2019). In general, explicit models define a parametric function over observed random variables in the data (e.g., features in tabular data), on which a tractable or approximate likelihood estimation can be performed to learn the underlying probability distribution. Contrariwise, likelihood estimation is not possible for implicit models since these models learn the probability distribution of data by a stochastic procedure (Mohamed & Lakshminarayanan, 2016). This procedure includes iteratively updating the parameters of a flexible transformation system (i.e., a neural network) that transforms noise, sampled from latent variables, into synthetic data records (Wu et al., 2019).

Where autoregressive and flow-based models are defined as explicit models, can DLVMs be explicit or implicit (Maaløe et al., 2019). In this research, both an explicit and an implicit model will be used to learn the real estate transaction data. To keep a tight scope and to assess a comparison between the performance of an explicit and an implicit model, only deep latent variable models will be taken into account from now on.

DLVMs make use of latent variables to model high-dimensional data. Latent variables are unobserved and commonly move in a lower-dimensional space (i.e., latent space) than the data. In (Chang, 2018), latent variable modeling in DGMs is discussed. This research states that for particular DLVMs, latent variables inferred from the data during the train process are the essence for usefulness and power.

Examples of explicit and implicit DLVMs are the VAE and GAN, respectively. These models, first proposed by (Kingma & Welling, 2013) and (Goodfellow et al., 2014), respectively, know many variations. One thing to note is that the VAE differs from other explicit models since it is characterized by a (posterior) distribution specified over the latent variables. This distribution is typically intractable and can only be approximated (Maaløe et al., 2019). Variational inference is a method in machine learning that can be used to approximate an intractable probability distribution (Blei et al., 2017).

Related Work

For a variety of applications, research has been conducted about how high-quality synthetic data can be generated. Examples are (Dahmen & Cook, 2019), who researched how to generate synthetic behavior-based sensor data for testing machine learning techniques used in healthcare applications. In addition, (Assefa, 2020) conducted a study about how synthetic data could be developed for applications in the financial sector and (Krishnan & Jawahar, 2016) researched how a synthetic image generator could create images containing handwritten text to train a deep neural network for text recognition.

Various DGMs can be used to learn the probability distribution of data and to generate synthetic data from that distribution. The quality of synthetic data heavily depends on the type of DGM, it's configuration and the learning process. To obtain a comprehensive overview of the quality of synthetic data (and indirectly of how well a DGM was able to learn a probability distribution), (Bourou et al., 2021) proposes to conduct a statistical analysis and a machine learning efficacy analysis on synthetic data. Before getting into related work regarding these analyses, existing literature on DGMs, used for synthetic data generation, is discussed.

5.1 Deep Generative Models on Homogeneous Data

Homogeneous data, such as pixels in images and words in text (Shwartz-Ziv & Armon, 2022) consists of values that originate from a single marginal distribution (Nazabal et al., 2020). VAEs and GANs have shown outstanding performances on learning probability distributions of homogeneous data such as graphs, natural text, and images.

Initially, (Goodfellow et al., 2014) proposed GANs to model the probability distribution of images, whereafter synthetic images could be sampled from that distribution. Hereafter, much complementary research is conducted about the use of GANs in computer vision. (Radford et al., 2015) introduced a deep convolutional neural network-based GAN architecture (DCGAN) and showed its applicability of generating good image representations that can be used for a variety of supervised deep learning tasks. Additionally, (Frid-Adar et al., 2018) and (Kukreja et al., 2020) investigated how GANs could be used to generate synthetic images to augment train data and increase the performance of supervised learning tasks such as image classification and recognition.

For a medical image classification task, (Frid-Adar et al., 2018) concluded that augmenting train images with synthetic images, generated by a GAN, resulted in a performance of 85.7% sensitivity and 92.4% specificity. This was an increase of 6.1% sensitivity and 4.0% specificity with respect to using classic data augmentation, in which the train images are augmented with rotated, scaled, flipped or sheared existing images. In addition, (Kukreja et al., 2020) augmented train images with by a GAN generated synthetic images to train a convolutional neural network that recognizes vehicle number plates. This resulted in an accuracy of 99.39%, which is 2.24% higher than the next-best performing vehicle number plate recognition model, proposed by (Omar et al., 2020).

Furthermore, synthetic data appears useful in natural language processing. (Bowman et al., 2015) show how a VAE can be used to model the probability distribution of text and generate synthetic natural language sentences that are coherent and diverse, making them realistic. Thereby, (Malandrakis et al., 2019) have exploited the VAE architecture to generate synthetic text in order to augment text train data for an intent classification task, which resulted in an F1-score performance increase of 5%.

Besides using synthetic data explicitly to augment the size of train data, synthetic data can also be used to balance imbalanced data. Data is imbalanced if there are great differences in the distribution of classes. In specific areas, it is crucial to find patterns from imbalanced data, such as health service, cyber security, and financial engineering (Wan et al., 2017). Undesired scenario's such as rare tumors on CT-scans, unusual network traffic and fraudulent transactions often represent a strong minority in data (Bhatia, 2019). Despite supervised deep learning techniques outperforming machine learning techniques when learning from large amounts of data, their performance decreases when learning from a large amount of data that is imbalanced (Lee & Park, 2021). Therefore, synthetic data could be helpful to balance the train data by augmenting the data with synthetic data containing instances for the minority class(es).

(Wan et al., 2017), (Bhatia, 2019) and (Lee & Park, 2021) show how VAEs and GANs can be used to augment imbalanced data with synthetic data. Although earlier methods, such as synthetic minority oversampling techniques (SMOTE), have demonstrated excellent performance coping with imbalanced data, they are undervalued to generative models like VAEs and GANs. Figure 5.1 shows the comparison of four performance metrics for a random forest classification model after augmenting the train data with synthetic data resampled by SMOTE and synthetic data generated by a GAN, respectively. It appears that augmenting the train data with synthetic data generated by the GAN results in a higher value for each performance metric of the random forest compared to augmenting the train data with synthetic data sampled by SMOTE.

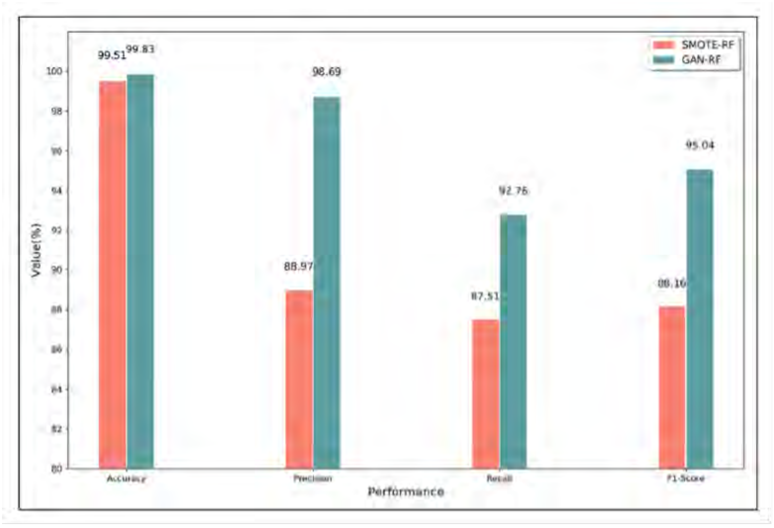


Figure 5.1: The classification performance metrics accuracy, precision, recall and F1-score for a Random Forest classifier after solving data imbalance by augmenting the train data with synthetic data. The orange and turquoise bars show the performance per metric after augmenting the train data with synthetic data sampled by SMOTE and after augmenting with synthetic data generated by a GAN, respectively (Lee & Park, 2021).

5.2 Deep Generative Models on Heterogeneous Data

Tabular data, which is nowadays the most encountered type of data in real-world applications (Shwartz-Ziv & Armon, 2022) may be heterogeneous. Heterogeneous tabular data consist of values that do not originate from a single marginal distribution. This could be due to the data consisting of a combination of mixed type features (i.e., a combination of continuous and categorical features) (Nazabal et al., 2020) or features of the same type but originating from different marginal distributions. From now, tabular heterogeneous data will be written as tabular data.

The development of DGMs such as GANs and VAEs, with their many extensions, have been very appealing because of their performance and flexibility in modeling data (Xu et al., 2019). Therefore, several studies are conducted in recent years about how to model tabular data with these models. An example is medGAN, proposed by (Choi et al., 2017), which combines an autoencoder and a GAN to model and generate high dimensional discrete (e.g., binary and categorical) electronic patient records.

(Xu et al., 2019) state that a tabular data set contains of N_c continuous features $\{C_1, ..., C_{N_c}\}$ and N_d categorical features $\{D_1, ..., D_{N_d}\}$, where all features are assumed to be random variables. This means that each record in tabular data $r_i = \{c_{1,i}, ..., c_{N_c,i},\ d_{1,i}, ..., d_{N_d,j}\},\ i \in \{1, ..., n\}$ and n being the total number of records, is an observation of the unknown joint probability distribution $\mathbb{P}(C_{1:N_c}, D_{1:N_d})$. When training a deep generative model on a particular tabular data set, the aim is to learn this unknown probability distribution and to generate synthetic data from it.

Challanges arise when learning the joint distribution of tabular data. In general, neural networks can effectively generate values with a centered distribution over the range $(-1, 1)$ (using $tanh$) and multinomial distributions (using $softmax$) (Xu & Veeramachaneni, 2018). However, continuous features in tabular data may be multimodal and non-Gaussian and therefore not centered over the interval $(-1, 1)$. Thereby, categorical features are initially not represented as a multinomial distribution. Therefore, both feature types need to be transformed into the desired format to enable a deep generative model to learn the probability distribution of tabular data properly.

The authors (Xu & Veeramachaneni, 2018), who proposed the TGAN model, introduced a reversible transformation method to convert heterogeneous data into the desired format. First, this method models multimodal continuous features with use of a Gaussian Mixture Model, whereafter the values are normalized into the range $(-1, 1)$. This technique is referred to as "mode-specific normalization". Categorical features are converted into one-hot encoded representations, whereafter noise is added. After training the TGAN model succesfully on the transformed tabular data, the model is able to generate synthetic data. The output is transformed back into the initial format of the tabular data by executing the transformation method in reversed order.

CTGAN, proposed by (Xu et al., 2019), is a follow up on the work of TGAN. Their mode-specific normalization method uses a variational Gaussian mixture model to estimate the number of modes in the distribution of a continuous feature instead of specifying this as a constant up front like for TGAN. Thereby, the CTGAN model uses a conditional generator an training by sampling to enable the model to learn highly imbalanced categorical features. An additional deep generative model, TVAE, was proposed to challenge the performance of CTGAN.

The performance of CTGAN, based on an appropriate metric, was compared against two baseline models using Bayesian Networks. In addition, the performances of three earlier proposed GAN models for tabular data modeling are compared to these baseline models. Table 5.1 shows the results. It appears that the CTGAN model outperforms the Bayesian networks on at least 7 out of 8 data sets. Thereby, the CTGAN model outperforms all earlier proposed methods.

Method	**CLBN**	**PrivBN**
MedGAN (2017)	1	1
VeeGAN (2017)	0	2
TableGAN (2018)	3	3
CTGAN (2019)	**7**	**8**

Table 5.1: The number of wins for four recently proposed methods with respect to the corresponding Bayesian networks (CLBN and PrivBN) based on an appropriate metric on 8 real data sets.

5.3 Quality Analysis

The quality of synthetic data depends on how well the DGM was able to learn the probability distribution from which the empirical data originates. However, a DGM can consist of millions of parameters and is therefore hard to interpret as a quality indicator.

Alternatively, the quality of a DGM can be assessed by comparing results of identical analyses on empirical data and synthetic data (El Emam, 2020). Two methods for analysis, proposed by (Bourou et al., 2021), are a statistical analysis and a machine learning efficacy analysis. The combination of these analyses gives a clear picture of the quality of the synthetic data and indirectly of how well the generating model was able to learn the probability distribution from which the empirical data originates.

5.3.1 Statistical Analysis

Statistical Tests

Using statistical tests is a quantitative approach to evaluate the quality of synthetic data feature-wise (Borisov et al., 2021). Statistical tests are developed to draw conclusions about a population based on a sample from that distribution (Lehmkuhl, 1996). There are two types of statistical tests, parametric and non-parametric tests.

To use a parametric test, a sample is assumed to be drawn from a known particular distribution (Lehmkuhl, 1996). Parametric tests use sample parameter values to calculate the difference relatively to the parameter values of the known distribution. Examples of parametric tests are the Student's T-test, Z-test, and Analysis of Variance (ANOVA) (Kaur & Kumar, 2015). Parametric tests could be used to test if a synthetic sample from a particular feature is drawn from the same distribution of its corresponding empirical feature for which the distribution in known.

On the other hand, if the distribution of the particular feature in the empirical data is unknown, a non-parametric tests might be valid. This type of test is distribution-free, and its structure is determined by the data (Kaur & Kumar, 2015). For example, the (two-sample) Kolmogorov-Smirnov test (Massey Jr, 1951) is appropriate to analyse if a sample from a continuous synthetic feature and corresponding empirical feature originate from the same distribution (Berger & Zhou, 2014; Bourou et al., 2021; Wen et al., 2021). In addition, the non-parametric Chi-Squared test of independence is a practical test to analyze if a synthetic categorical feature is independent of the corresponding empirical feature (Bourou et al., 2021; Connelly, 2019; McHugh, 2013; Wen et al., 2021).

However, when it comes to big data, statistical power is a concern since traditional significance testing might be less relevant than for relatively small samples. Statistical tests using large samples might consider trivial differences to be statistically significant (Bergh, 2015). Therefore, using large samples and relying only on p-values and test statistics to evaluate the concordance between two large data samples might lead to false conclusions. This is referred to as "The large sample size fallacy" (Lantz, 2013).

Correlation

High quality synthetic data resembles the correlations for all feature combinations in empirical data (Bcom, 2012). The method to calculate the correlation between two features depends on their types.

First, to calculate the correlation between a synthetic continuous feature and its corresponding empirical continuous feature, Pearson's Correlation Coefficient is appropriate (Baak et al., 2020). This claim is substantiated by (Baak et al., 2020; Bcom, 2012), who claim that Pearson's correlation coefficient is the most commonly used coefficient for continuous features and even a "de facto" in many fields.

Second, the Pearson Chi-squared test of independence is used to determine if two categorical features are independent. If the null hypothesis (two samples are independent) is rejected, the strength of the correlation can be measured with Cramer's V, a normalized Chi-squared value (Eckelt et al., 2019). Cramer's V is a variation of the Phi-Coefficient. This coefficient measures the correlation between two categorical features, both having a maximum cardinality of two. Cramer's V can be used for categorical features with higher cardinality (Akoglu, 2018).

Another measurement for correlation between categorical features is Theil's U (Brenninkmeijer & Hille, 2019; Shannon, 1948). In contrast to Cramer's V, Theil's U takes the conditional entropy of both categorical features into account. The conditional entropy quantifies the uncertainty to describe the outcome of one feature, given the value the another feature (Walters-Williams & Li, 2009).

Third, to calculate the correlation between a categorical and a numerical feature, the Correlation Ratio can be used (Fisher, 1970). This ratio is defined as the weighted variance for each category mean, divided by the variance of the continuous feature (Brenninkmeijer & Hille, 2019).

Likelihood

Instead of evaluating the quality of synthetic data feature-wise, the probability distribution of a generated synthetic sample can be compared with the ground truth (i.e., the probability distribution from which the empirical data originates) in terms of log-likelihood (Wen et al., 2021; Xu & Veeramachaneni, 2018). This is done by regarding a Bayesian Network (BN) as oracle that is used to generate the empirical data. Hereafter, by taking advantage of the oracle model, the log-likelihood of the synthetic data is calculated. The higher this value, the more likely the synthetic data is generated from the same distribution as the empirical data.

Privacy

A commonly used metric to indicate the level of privacy of synthetic data is Distance to Closest Record (DCR). This metric calculates the euclidean distance between a record in the synthetic data and the closest record in the empirical data (Park et al., 2018). A value of 0 indicates that the synthetic record and the closest empirical record have a Euclidean distance of 0. In other words, the two records are equal, and therefore the synthetic record leaks information from the empirical data.

Similarity score

To give an overall indication of the quality of synthetic data, (Brenninkmeijer & Hille, 2019) proposed a similarity score. This metric returns the average quality over five different statistical metrics, which are the following:

1. Basic measures: Correlation between means, medians, standard deviations, and variances for continuous features in synthetic and empirical data.
2. Correlation correlations: Pearson's r correlation coefficient between correlations in synthetic and empirical data.
3. Mirror correlations: Pearson's r correlation coefficient between corresponding features in synthetic and empirical data.
4. PCA: Explained variance of N principal components for synthetic and empirical data.
5. Machine Learning efficacy: Performance measures for various supervised learning algorithms trained on synthetic data and tested on an unseen empirical test set. These performances are compared with the performance of the same models trained on empirical data.

The mean value of element $1 - 5$ represents the similarity score and lies in the range $[0, 1]$ with 1 indicating high similarity and vice versa.

5.3.2 Machine Learning Efficacy Analysis

Research on synthetic data quality by evaluating the performance of a supervised learning model, trained on synthetic data and tested on unseen empirical test data, gives better insight into how well the deep generative model generalizes and approximates the unknown joint probability distribution from which the empirical data originates.

Detection

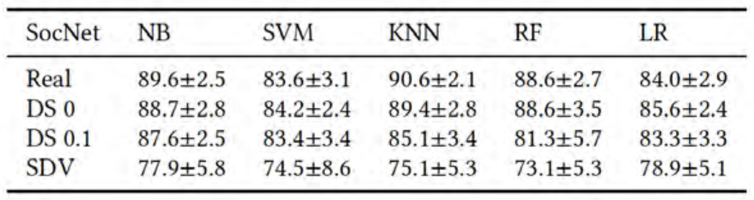
One way to evaluate the quality of synthetic based on machine learning is by using a detection metric. This metric indicates how difficult it is for a classification algorithm to distinguish between empirical and synthetic data records. The empirical records and the synthetic records are flagged with a 1 and 0, respectively. Hereafter, the flagged data is shuffled and a classifier is cross-validated, aiming to predict the flag. The number of true positives (i.e., correctly labeled empirical records and false positives (i.e., incorrectly labeled synthetic records) determines the quality of the synthetic data. A high percentage of true positives and false positives indicates that the classifier finds it hard to distinguish the real records from the synthetic records (Bourou et al., 2021).

Machine Learning Efficacy

Synthetic data could be used as training data instead of empirical data for supervised machine learning models (Hittmeir et al., 2019). The performance difference between models trained on synthetic data and models trained the on empirical data can be used as a metric for synthetic data quality (Bourou et al., 2021; Park et al., 2018; Xu et al., 2019; Xu & Veeramachaneni, 2018).

(Borisov et al., 2021) even states that the most common way to assess synthetic data quality is to define a classification task, train classification models separately on empirical and synthetic data and compare their performance on an unseen empirical test set.

An example is the comparison of accuracy performances for five different classifiers in (Hittmeir et al., 2019). The classifiers are trained ten times on empirical data and synthetic data, generated by three different models (DS 0, DS 0.1 and SDV). For the classifiers, NB (Naïve Bayes), SVM (Support Vector Machine), KNN (K-Nearest Neighbor), RF (Random Forest), and LR (Logistic Regression), the largest mean absolute differences in accuracy are 11.7, 9.1, 15.5, 15.5, and 5.1 respectively. Table 5.2 shows the results.



SocNet	NB	SVM	KNN	RF	LR
Real	89.6±2.5	83.6±3.1	90.6±2.1	88.6±2.7	84.0±2.9
DS 0	88.7±2.8	84.2±2.4	89.4±2.8	88.6±3.5	85.6±2.4
DS 0.1	87.6±2.5	83.4±3.4	85.1±3.4	81.3±5.7	83.3±3.3
SDV	77.9±5.8	74.5±8.6	75.1±5.3	73.1±5.3	78.9±5.1

Figure 5.2: Mean accuracy scores with standard deviation for five different classifiers, trained 10 times separately on empirical and synthetic data samples (Hittmeir et al., 2019).

The results show that the classifier models trained on synthetic data have a decrease in performance compared to the models trained on empirical data, however, the differences are small. In addition, (Rankin et al., 2020) state that the analytical potential in healthcare data, which is too sensitive to share, could be unlocked with synthetic data. In their research, five supervised machine learning models were trained separately on empirical and synthetic data and tested on empirical data. Despite models trained on synthetic data have a decrease of accuracy, the differences are small and manageable.

Data Description and Preparation

This chapter includes information on the available data and is divided into three sections. The first section, Data Description (6.1), introduces the reader to the real estate transaction data. Hereafter, the exploration of the data is discussed in section 6.2, followed by the applied pre-processing steps, including making the data suitable as input for generative models, which are discussed in section 6.3.

6.1 Data Description

Pararius is a company that offers the most prominent independent housing platform in the Netherlands, focusing on rental properties in the private sector. Various parties, such as real estate agents and individuals, can register properties manually on the platform. Every time a registered rental property is rented out to a customer, the data about this transaction is stored. All available transactions from Pararius together form the dataset for this research and are made in the period 01/01/2021 – 01/07/2021. However, the transactions in the data are not listed chronologically, which means that there is no time aspect to take into account.

6.2 Data Exploration

The data, known as Rental Transaction Data (RTD), include 87.389 transactions consisting of 17 features. The party that registers the rental property at the housing platform provides the data about the property, which consists of continuous, categorical, and binary features. Table 6.1 gives an overview of the transaction features, their corresponding types, the number of unique values (cardinality), and a brief description.

Feature	**Type**	**Unique values**	**Description**
listing price	continuous	-	The rental price per month for the listed property
listing size	continuous	-	The size of the property in square meters
parking availability	binary	2	Indicator whether there is parking availability
garden	binary	2	Indicator whether a garden is present
storage	binary	2	Indicator whether a storage is present
garage	binary	2	Indicator whether a garage is present
balcony	binary	2	Indicator whether a balcony is present
floor level	ordinal	31	The floor level at which the property is located
number of floors building	ordinal	6	The total number of floors of the building in which the property is located
total rooms	ordinal	19	Total number of rooms in the listed property, excluding bathroom and toilet
total bedrooms	ordinal	13	Total number of bedrooms in the listed property
maintenance status	ordinal	5	The maintenance status
energy label	ordinal	12	The energy label
residential type	categorical	10	The type of property
interior type	categorical	3	The interior type of the property
parking type	categorical	6	The type of parking availability, in case there is a parking availability
postal code	categorical	35.868	The postal code in which the property is located

Table 6.1: The features present in the RTD, including feature types, number of unique values and a brief description.

6.2.1 Missing Values

The RTD is sparse due to several categorical and binary features containing a significant percentage of missing values. Furthermore, since the RTD is provided by humans, filling in the feature values when a property is registered is not a homogeneous process per se. Therefore, the meaning of a missing value may be ambiguous.

A missing value could indicate missing information about the feature or that the feature is not present for a particular property (e.g., there is no information about the property having a garage, or there is no garage present at the property). In the latter case, the party who registered the property and left the garage feature value empty provided data incorrectly and should have inserted 0, indicating no garage available. Nevertheless, since in this research it is desired to modify the data as little as possible to stay as close as possible to reality, missing values are kept as missing values, and no imputation methods are used. Table 6.2 gives the percentage missing values per feature.

Feature	**% missing values**	**Feature**	**% missing values**
listing price	00.0010%	total rooms	00.00%
listing size.	00.0035%	total bedrooms	02.96%
parking availability	66.69%	maintenance status	61.30%
garden	62.13%	energy label	58.61%
storage	59.25%	residential type	00.00%
garage	62.47%	interior type	06.90%
balcony	65.01%	parking type	60.90%
floor level	60.53%	postal code	00.00%
number of floors building	57.96%	-	-

Table 6.2: Percentage of missing values per feature in the RTD.

6.3 Pre-processing

Since this research investigates how well generative models can learn the probability distribution from which the raw RTD originates, it is desired to keep the data as original as possible and modify it as little as possible.

However, to make the data as complete as possible and suitable as input for generative models, publicly available data is used to extend the RTD with additional features, followed by some steps to clean and modify the data. First, subsection 6.3.1 describes the feature engineering process. Hereafter, subsections 6.3.2 and 6.3.3 describe handling invalid values and extreme values, respectively.

6.3.1 Feature Engineering

In machine learning, learning a “state-of-nature” from a finite number of data records requires a large amount of training data. This is required since, for a model to learn patterns in data, a significant amount of records reflecting the pattern must be available for a model to recognize it. The required number of records increases with the number of features and the cardinality of categorical features. The Hughes phenomenon (Hughes, 1968) states that the predictive power for a supervised learning algorithm (e.g., a regressor or classifier) increases when parameters, such as the number of features or feature cardinality, increase. However, its power deteriorates when these parameter values become too large. The Hughes phenomenon is used as a guide for the feature engineering of the RTD.

Clappform possesses a publicly available dataset which contains the features postal code with corresponding neighborhood, and municipality features. The latter two have a significantly lower cardinality than the postal code feature. Therefore, the publicly available data set and the RTD are merged on the feature postal code. Hereafter, the postal code feature is removed. The number of features in the data increased by one, but the maximum cardinality is reduced from 35.868 to 5.856. Table 6.3 gives an overview of the added features.

Feature	**Type**	**Unique values**	**Description**
neighborhood	categorical	5.856	The neighborhood in which the listed property is located
municipality	categorical	349	The municipality in which the listed property is located

Table 6.3: Added features with their type, number of unique values and a brief description.

6.3.2 Invalid Values

First, the data is checked on duplicate transactions. In total, 662 duplicates were found and removed. This leads to a resulting data set containing 86.727 transactions.

Second, it might be the case that a value is ambiguous or invalid. The values -1 and “unknown” indicate that the value is unknown (this statement is based on the domain knowledge of the data analytics team of Pararius). Since these values have the same meaning as a missing value, they are transformed into a missing value. This transformation leads to a new total percentage of missing values for the concerned features. Table 6.4 shows the features containing invalid or ambiguous values and their final percentage of missing values after transforming invalid and ambiguous values into a missing value.

Feature	Invalid value	Final % missing
maintenance status	-1	92.22%
energy label	-1	89.53%
parking type	-1	91.82%
interior type	”unknown”	10.86%

Table 6.4: Final percentage missing values after transforming invalid values into missing values.

In addition, some feature combinations are subject to constraints. For example, the feature “parking type” can’t have any value when ”parking availability” is equal to 0. Likewise, in case there is a value registered for feature ”numbers of floors building”, the value of feature “floor level” cannot exceed. The same condition holds for ”total bedrooms” in case there is a value registered for ”total rooms”. Table 6.5 shows the feature combinations that are subjective to constraints together with the number of records in the RTD that did not meet the constraints .

Feature 1	Feature 2	Invalidness	Count
parking type	parking availability	Feature 1 contains value & Feature 2 = 0	0
floor level	number of floors building	floor level > number of floors building	4.076
total rooms	total bedrooms	total rooms < total bedrooms	76

Table 6.5: Feature combinations which are subject to constraints, the rule when a constraint is violated and the number of records violating the constraint.

All transactions that contain invalid feature combinations are modified. Since there are no invalid records for the feature combination ”parking type” and ”parking availability”, only the feature combinations “floor level” together with “number of floors building” and “total rooms” together with “total bedrooms” are modified. Removing the records with invalid value combinations from the RTD would directly reduce the number of values from other features that already have a high percentage of missing values. Instead, the values within the feature combinations are switched. This means that for invalid records, the “total bedrooms” feature takes the value of the “total rooms” feature and vice versa. The same is done for “floor level” and “number of floors building”. The decision to switch values is discussed with the data analytics team of Pararius. They argued that parties who register properties could be confused and inaccurate when providing data about these feature combinations.

Finally, the “neighborhood” and “municipality” features consist of unique combinations since a neighborhood simply cannot change to a different municipality. These features are used to merge the RTD with the publicly available data. The merged data will later be used as input for Clappform’s rental price prediction model. This means that these features cannot contain any missing values and are therefore removed if they do. In total, 537 records contained a missing value for one of the features ”neighborhood” or ”municipality” and are removed. This leads to a remaining data set containing 86.190 records

6.3.3 Extreme Values

Extreme values, or outliers, are caused by two phenomena (Hoogendoorn Funk, 2018). These phenomena are:

- Measurement errors
- Variability in the observed feature

An outlier can be defined as “an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism” (Hawkins, 1980). In addition, outliers can influence the predictive power of regression models, classification models (Kalisch et al., 2015), and empirical model identification in general (R. K. Pearson, 2002). For both reasons, it is often desired to remove outliers from data.

However, this research aims to investigate how well generative models can learn the probability distribution and reproduce characteristics of the RTD. That includes studying how well a generative model can learn extreme values, caused by variability, in the data. Still, outliers caused by measurement errors should be removed. One approach is by consulting experts with domain knowledge. Another method is machine learning (Hoogendoorn & Funk, 2018). Since Pararius and Clappform both have data analytics experts with domain knowledge about the RTD, threshold values are set manually based on their expertise and advise, whereafter outliers are removed.

Categorical features have predefined categories without ordering. Therefore, outlier detection and removal is only applicable to continuous and ordinal features. These features are: “listing price”, “listing size”, “floor level”, “number of floors building”, “total rooms” and “total bedrooms”. Table 6.6 shows for each feature the threshold value. All values above these thresholds are removed from the data. In total, 83 outliers are removed, resulting in a data set with 86.007 transaction records.

Feature	**Threshold value**
listing price	7500
listing size	600
floor level	6
number of floors building	12
total rooms	10
total bedrooms	8

Table 6.6: Threshold values for features containing outlier values.

Methodology

To learn real estate transaction data, two deep latent variable models are trained to learn the underlying probability distribution of the RTD. Statistical methods will be used to analyze how well the deep latent variable models learned the *empirical* probability distribution by quantifying how well the models reproduced statistical properties of the RTD in synthetic samples. Thereby, Clappform's real estate rental price forecasting model (ERV model) is trained on synthetic samples and tested on the empirical test set to quantify how well the deep generative models learned the underlying distribution of the RTD. This will be the machine learning efficacy analysis. The deep generative models and the methods used in the analyses are discussed in this chapter.

First, two fundamental deep generative algorithms, VAE and GAN, are explained in subsections 7.1.2 and 7.1.1, respectively. Hereafter, the (open source) Python package, Synthetic Data Vault (SDV), is discussed in section 7.2, with the reversible data transformation that is applied on the RTD in section 7.2.1 and the selected deep latent variable models, CTGAN (7.2.2) and TVAE (7.2.3). In section 7.3, 7.4 and 7.5, methods are described that are used in the statitiscal and machine learning efficacy analysis. Finally, in section 7.6, the fundamental regression algorithm of the ERV model is discussed

7.1 Deep Latent Variable Models

Learning complicated high dimensional probability distributions, $p(x)$, from a finite number of samples (x) might be challenging (Chen et al., 2018). Instead of modeling $p(x)$ directly, deep latent variable models use latent variables (z). In this section, two deep latent variables models are discssed. First an explicit model in subsection 7.1.1, the Variational Autoencoder. Hereafter, an implicit model, the Generative Adversarial Network, is discussed in subsection 7.1.2.

7.1.1 Variational Autoencoder

A variational autoencoder (VAE) consists of two neural networks (i.e., multilayer perceptrons), the encoder network and the decoder network. The input data (x) is given to the encoder network. This network compresses x into the (lower dimension of) the latent space. The latent space consists of parameters for distributions that explicitly specify the latent space (Blei et al., 2017) (e.g. means (μ) and standard deviations (σ) for a multivariate Gaussian distribution). Hereafter, the decoder network takes a sample from the latent space and decompresses it back into the original dimension of the input data, aiming to output a reconstruction of the input data. Finally, the error is calculated between the input data and the reconstructed output data, whereafter backpropagation is executed. During backpropagation, both the parameters of the encoder and the decoder are adjusted. In Figure 7.1, an example of a VAE with a two dimensional latent space is shown.

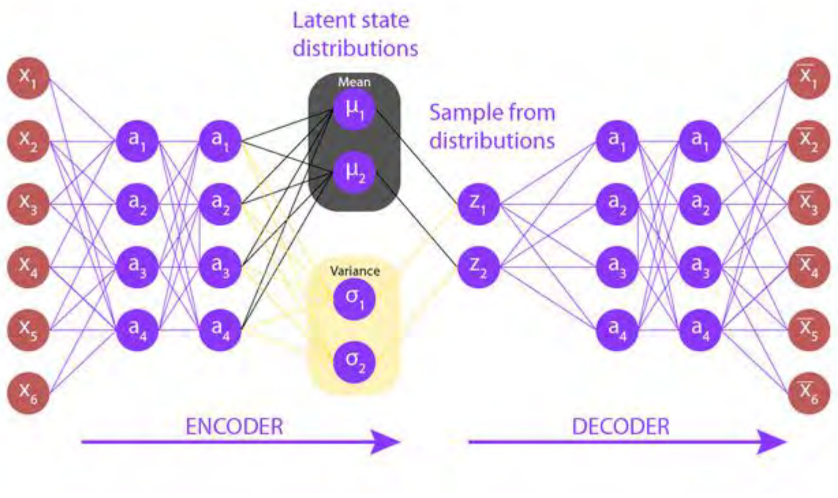

Latent state distributions
Mean
Variance
Sample from distributions
ENCODER
DECODER
x_1 x_2 x_3 x_4 x_5 x_6
a_1 a_2 a_3 a_4
μ_1 μ_2
σ_1 σ_2
z_1 z_2
$\bar{x}_1$ $\bar{x}_2$ $\bar{x}_3$ $\bar{x}_4$ $\bar{x}_5$ $\bar{x}_6$

Figure 7.1: The architecture of a VAE with a two dimensional latent space.

In Bayesian statistics, a prior distribution can be described as "the assumed to be true" distribution of random variable(s) before any observed data about the random variable(s) is taken into account. Contrariwise, the posterior distribution is the distribution over random variable(s), given some data. The explicitly specified distribution over the latent variables z, which characterizes a VAE after being trained, is the posterior distribution. With use of Bayes' theorem, the posterior distribution over z given x can be written as:

$$p_\theta(z|x) = \frac{p_\theta(x,z)}{p_\theta(x)} = \frac{p_\theta(x|z)p(z)}{p_\theta(x)} \tag{7.1}$$

With $p_\theta(x|z)$, $p_\theta(z)$, $p_\theta(x)$ and θ being the likelihood of x, given z, the prior distribution over z, the marginal probability distribution of x and the network parameters, respectively. Using the likelihood and the prior distribution, $p(x)$ can be written as:

$$p_\theta(x) = \int p_\theta(x|z)p_\theta(z)dz = \int p_\theta(x,z)dz \tag{7.2}$$

An attempt to compute $p_\theta(x)$ in equation 7.2 could be done by marginalizing over the latent variables. However, doing this for all possible values for z might require exponential time. Therefore, $p_\theta(x)$ is often intractable (Maaløe et al., 2019). This means that the posterior in equation 7.1 is also intractable. Therefore, $p_\theta(x)$, and thus $p_\theta(z|x)$, can only be approximated. One way to approximate $p_\theta(z|x)$ is by using variational inference, a method to approximate probability distributions by optimization (Blei et al., 2017).

Formally, the encoder network can be defined as the (variational posterior) distribution $q_\phi(z|x)$, used to approximate the intractable true posterior distribution $p_\theta(z|x)$, and the decoder network as the likelihood $p_\theta(x|z)$. The network parameters in the encoder and decoder are are ϕ, and θ, respectively. The aim is to train the VAE in such a way, that the distributions $q_\phi(z|x)$ and $p_\theta(z|x)$ are as similar as possible. To measure the distance (or dissimilarity) between distributions, the Kullback-Leibler divergence can be used. Using this divergence, the marginal likelihood can be written as

$$log(p_\theta(x)) = D_{KL}(q_\phi(z|x) \,||\, p_\theta(z|x)) + E_{z\sim q_\phi(z|x)}(log(p_\theta(x|z))) - D_{KL}(q_\phi(z|x) \,||\, p_\theta(z)) \tag{7.3}$$

Equation 7.3 can be split into two components, the Kullback-Leibler divergence between $q_\phi(z|x)$ and $p_\theta(z|x)$) (1) and the evidence lower bound (ELBO) (2).

1. $D_{KL}(q_\phi(z|x) \,||\, p_\theta(z|x))$
2. $E_{z\sim q_\phi(z|x)}(log(p_\theta(x|z))) - D_{KL}(q_\phi(z|x) \,||\, p_\theta(z))$

Equation 7.3 can be rewritten as:

$$log(p_\theta(x)) - D_{KL}(q_\phi(z|x) \,||\, p_\theta(z|x)) = E_{z\sim q_\phi(z|x)}(log(p_\theta(x|z))) - D_{KL}(q_\phi(z|x) \,||\, p_\theta(z)) \tag{7.4}$$

When training a VAE, the aim is to maximize the left hand side of equation 7.4 (Wan et al., 2017). Since the true posterior $p_\theta(z|x)$ is intractable, the Kullback-Leibler divergence cannot be calculated. However, it is known that this quantity is non-negative. This implies that $log(p_\theta(x)) >$ ELBO. Therefore, maximizing the left hand size of equation 7.4 is achieved by optimizing the ELBO with respect to the parameters ϕ and θ in the encoder and decoder, respectively (Kingma & Welling, 2013). This can be done with use of gradient descent.

By optimizing the ELBO, the approximate posterior is pushed towards the true posterior. If the true posterior is approximated well, the decoder network can be used to sample from the variational posterior in order to generate synthetic data originating from a distribution that is approximately equal to $p_\theta(x)$. In other words, the decoder network can be considered a generative model.

7.1.2 Generative Adversarial Network

A Generative Adversarial Network (GAN) is a framework for constructing generative models via an adversarial process. The framework consists of two neural networks (i.e., multilayer perceptrons), the generative model G and the discriminative model D, respectively. The generative model aims to capture the probability distribution from which the data is drawn, p_{data}, by learning an implicitly defined distribution p_g over the empirical data. The discriminative model estimates for a particular data sample how likely it is that the data sample originates from p_{data} by returning a probability value.

Formally, the generator can be defined as a differentiable function $G(\mathbf{z};\theta_g)$, with (network) parameters θ_g, that takes a latent space sample $\mathbf{z}$ as input and maps it into the data space. A prior distribution, $p(\mathbf{z})$ is specified to enable sampling from the latent space. Initially, (Goodfellow et al., 2014) used the uniform distribution to specify the distribution over the latent space. However, in other studies, the Gaussian distribution is commonly used (Brenninkmeijer & Hille, 2019)[1]. The discriminator can be defined as the differentiable function $D(\mathbf{x};\theta_d)$, with (network) parameters θ_d, that returns a probability.

Both models are trained simultaneously, based on a minimax two-player game. In a minimax game, the objective can be interpret as one player trying to maximize its probability of winning, whereas the second player is trying to minimize the winning probability of the first player. For a GAN, D is the first player and G is the second player. This means that the aim of the training procedure is to train G in such a way that D classifies a by G generated data sample incorrectly with high probability and to train D in such a way that it correctly classifies empirical and by G generated data samples with high probability. If both models are trained successfully and D returns a high probability for a by G generated data sample, this means that D considers this sample to be drawn from p_{data} and thus finds it hard to distinguish between empirical data samples and samples generated by G. Figure 7.2 shows an example of a GAN.

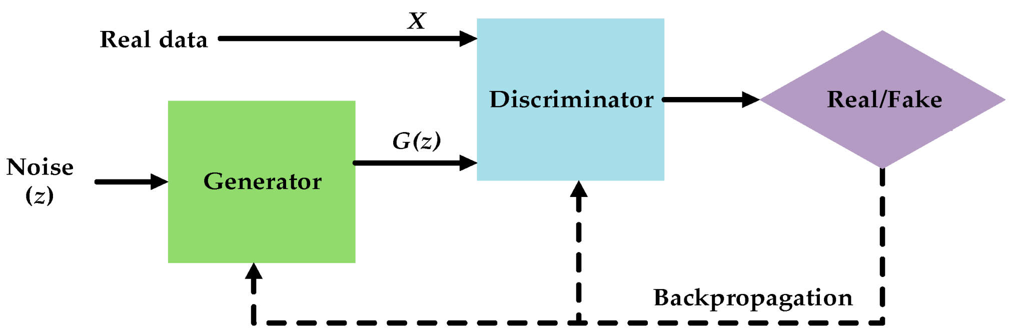


Figure 7.2: The architecture of a GAN

The training procedure, or minimax game, is defined by the following loss function, called the value function $V(G,D)$. Mathetmatically this function can be written as:

$$\min_G \max_D V(G,D) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[log(1-D(G(\mathbf{z})))] \tag{7.5}$$

Equation 7.5 consists of two terms:

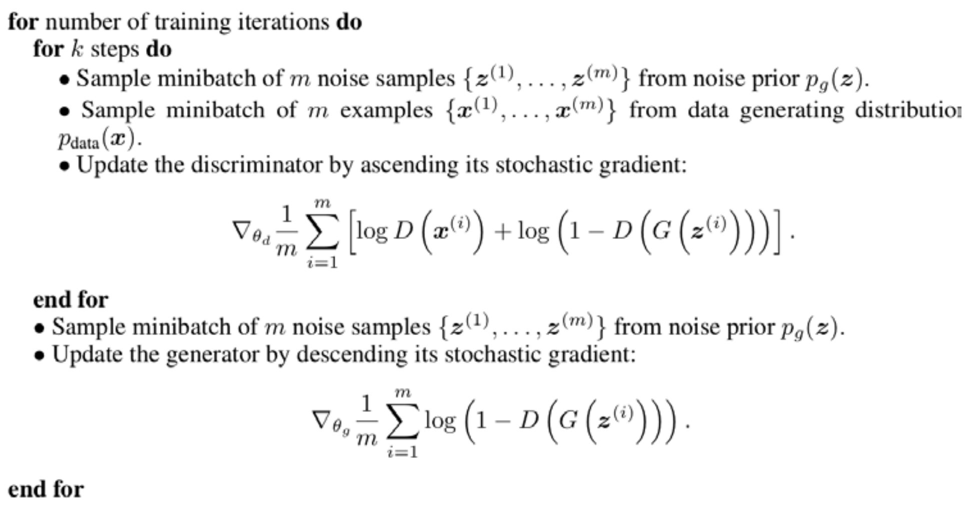
1. $\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[log(D(\mathbf{x}))]$
2. $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[log(1-D(G(\mathbf{z})))]$

In which $D(\mathbf{x})$ is the discriminator's probability estimate that empirical sample $\mathbf{x}$ originates from p_{data} and $D(G(\mathbf{z}))$ the discriminator's probability estimate that a by the generator created sample $G(\mathbf{z})$ originates from p_{data}. This means that the first and the second expectation term in equation 7.5 can be defined as the expected log probability that the discriminator correctly classifies empirical samples and the expected log probability that the discriminator correctly classifies generated samples, respectively.

What should be noticed is that the D appears in both terms whereas G only appears in the second term. This means that for the generator the only objective is to minimize the second term. However, for the discriminator, the objective is to maximize both terms. In addition, the order in which both objectives are executed is important. Iteratively, the parameters θ_d in D are updated by maximizing the first and the second term. Hereafter, the parameters θ_g in G are updated by minimizing the second term. The updating process of the

[1]The dimension of the latent space is set before training. When sampling from the latent space, typically a value is drawn from the standard normal distribution for each dimension. This research uses a latent space with dimension equal to 100 and samples from the standard normal distribution for each dimension.

network parameters is done by performing minibatch stochastic gradient descent. This iterative process is described more detailed in figure 7.3.



```
for number of training iterations do
    for $k$ steps do
        • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
        • Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distributio
        $p_{\text{data}}(\boldsymbol{x})$.
        • Update the discriminator by ascending its stochastic gradient:
            $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$
    end for
    • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Update the generator by descending its stochastic gradient:
        $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$
end for
```

Figure 7.3: The training algorithm for a GAN.

Finally, (Goodfellow et al., 2014) state that if D and G are trained successfully, the generator distribution p_g converges to p_{data}. In that case, G can be used to generate synthetic data that originates from the distribution p_{data}.

7.2 Synthetic Data Vault

The open source Synthetic Data Vault (SDV) package is an overall system for synthetic data models, benchmarks and metrics. The SDV contains models that are able to model (and generate synthetic) tabular data and time series data. The models for tabular data can be split into categories based on classical statistical methods (such as copulas) and models based on deep learning methods. Since the scope of this research only includes deep generative models, the latter category of models will be used to learn the RTD. This category includes the deep generative models CTGAN and TVAE, proposed by (Xu et al., 2019), and are discussed in subsections 7.2.2 and 7.2.3, respectively. Both models transform the heterogeneous RTD data before taking it as input. This method is discussed in subsection 7.2.1. Before continuing, some notation must be defined:

- $x_1 \oplus x_2 ...$: concatenate vectors $x_1, x_2, ...$
- $gumbel_\tau(x)$: apply Gumbel softmax (Jang et al., 2016) with parameter τ on vector x
- $leaky_\gamma(x)$: apply leaky ReLU activation function on vector x with leaky ratio γ
- $FC_{u \to v}(x)$: apply a linear transformation on a u-dimensional input to get a v-dimensional output
- *cond*: conditional vector

Thereby, BN stands for batch normalization and *drop* for dropout. Batch normalizing is a method to stabilize the neural network by normalizing the values in the hidden layers (Ioffe & Szegedy, 2015). Dropout is a regularization method to prevent neural networks from overfitting by randomly dropping nodes in the hidden layers (Srivastava et al., 2014).

7.2.1 Reversible Data Transformation

Mode-specific Normalization

The mode-specific normalization method is used to convert the values of a multimodal (non-Gaussian) continuous feature into the range $(-1, 1)$ by applying the following steps:

1. For each continuous feature C_i, a variational Gaussian mixture model (VGM) is used to estimate the number of modes m_i, whereafter a Gaussian mixture model is fitted. For example, in figure 7.4 the VGM finds three modes ($m_i = 3$), namely η_1, η_2 and η_3. The learned Gaussian mixture for column C_i is $\mathbb{P}_{C_i}(c_{i,j}) = \sum_{k=1}^{3} \mu_k \mathcal{N}(c_{i,j}; \eta_k, \phi_k)$ where μ_k and ϕ_k are the weight and the standard deviation of a mode, respectively.

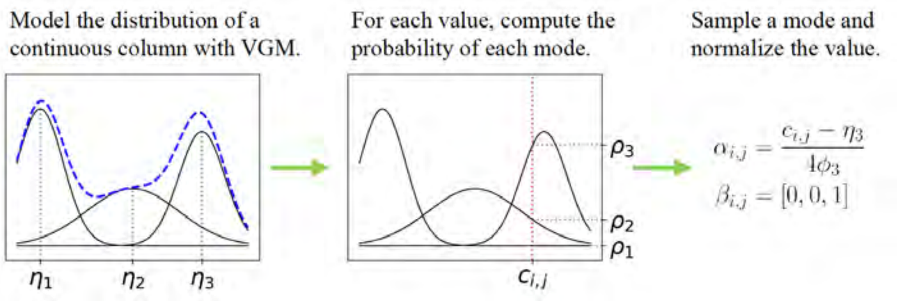

Model the distribution of a continuous column with VGM.
For each value, compute the probability of each mode.
Sample a mode and normalize the value.
η_1
η_2
η_3
ρ_3
ρ_2
ρ_1
$c_{i,j}$
$\alpha_{i,j} = \frac{c_{i,j} - \eta_3}{4\phi_3}$
$\beta_{i,j} = [0, 0, 1]$

Figure 7.4: An example of mode specific normalization

2. For each value $c_{i,j}$ in feature C_i, compute the probability of $c_{i,j}$ coming from each mode. For example, in figure 7.4, the probability densities are ρ_1, ρ_2 and ρ_3 and are calculated as $\rho_k = \mu_k \mathcal{N}(c_{i,j}; \eta_k, \phi_k)$.

3. Sample one mode and use this mode to normalize the value $c_{i,j}$ For example, in figure 7.4, the third mode is sampled. Then $c_{i,j}$ is represented by a one-hot encoded vector $\beta_{i,j} = [0, 0, 1]$, indicating the third mode, and a scalar $a_{i,j} = \frac{c_{i,j} - \eta_3}{4\phi_3}$.

Transforming Categorical Features

The values in categorical features are represented by one-hot encoded vectors. For each unique category, a new column is created. A value in this column is indicated with a one if the category value appeared in the original column, otherwise zero.

Finally, all transformed values for continuous and categorical features are concatenated together, which leads to the following representation for a row $\mathbf{r}_j$:

$$\mathbf{r}_j = \alpha_{1,j} \oplus \beta_{1,j} \oplus ... \oplus \alpha_{N_c,j} \oplus \beta_{N_c,j} \oplus \mathbf{d}_{1,j} \oplus ... \oplus \mathbf{d}_{N_d,j} \tag{7.6}$$

In which $\mathbf{d}_{i,j}$ represents the one-hot encoded vector for value j of categorical feature i (Xu et al., 2019). The vectors $\beta_{i,j}$ are included such that the the model knows which mode to use when transforming the continuous output back into the original format of the data.

7.2.2 CTGAN

Since the features in the RTD have no local structure, the generator and the discriminator model in the CTGAN are represented by two fully-connected neural networks to capture all possible correlations between the features. Both networks consist of two hidden layers, followed by the output layer. In the output layer of the generator network, values are generated by a mix of non-linear activation functions. The scalars $a_{i,j}$ are generated by applying *tanh*, while the one-hot encoded vectors $\beta_{i,j}$ and $\mathbf{d}_{i,j}$ are generated by the Gumbel softmax function. Formally, the structure of the generator network can be written as:

$$G(z, cond) : \begin{cases} h_0 = z \oplus cond \\ h_1 = h_0 \oplus ReLU(BN(FC_{|cond|+|z| \to 256}(h_0))) \\ h_2 = h_1 \oplus ReLU(BN(FC_{|cond|+|z|+256 \to 256}(h_1))) \\ \hat{\alpha}_i = tanh(FC_{|cond|+|z|+512 \to 1}(h_2)), \quad 1 \leq i \leq N_c \\ \hat{\beta}_i = gumbel_{0.2}(FC_{|cond|+|z|+512 \to m_i}(h_2)), \quad 1 \leq i \leq N_c \\ \hat{\mathbf{d}}_i = gumbel_{0.2}(FC_{|cond|+|z|+512 \to |D_i|}(h_2)), \quad 1 \leq i \leq N_c \end{cases} \tag{7.7}$$

The input of the discriminator is a concatenation of 10 samples and 10 conditional vectors. This is done to prevent the model from mode collaps. The architecture of the discriminator can formally be written as:

$$D(\mathbf{r}_1, ..., \mathbf{r}_{10}, cond_1, ..., cond_{10}) : \begin{cases} h_0 = \mathbf{r}_1 \oplus ... \oplus \mathbf{r}_{10} \oplus cond_1 \oplus ... \oplus cond_{10} \\ h_1 = drop(leaky_{0.2}(FC_{10|\mathbf{r}|+10|cond| \to 256}(h_0))) \\ h_2 = drop(leaky_{0.2}FC_{256 \to 256}(h_0))) \\ D(\cdot) = FC_{256 \to 1}(h_2) \end{cases} \tag{7.8}$$

The model is trained with use of the WGAN loss function with gradient penalty (Gulrajani et al., 2017) and the Adam optimzer.

Conditional Generator

The regular training process of a GAN does not account for highly imbalanced categorical features. In this process, records are randomly sampled. Therefore, records that belong to the minority class in an imbalanced feature might shrink into insignificance compared to the other records and will lead to an insufficient trained generator.

To solve this limitation, a generator of the CTGAN is adapted to a conditional generator. This generator is able to generate synthetic records that are conditioned on one of the discrete columns. In this way, the CTGAN can equally explore all possible values for each discrete feature. This results in highly imbalanced features in the empirical training data being reconstructed correctly in synthetic data.

7.2.3 TVAE

To create the TVAE model, a regular VAE model is adapted. The adapted model consists of an encoder network and a decoder (or generator) network to model $p_\phi(z_j|\mathbf{r}_j)$ and $p_\theta(\mathbf{r}_j|z_j)$, respectively. Both networks consist of two hidden layers, followed by the output layer. The model is trained by optimizing the ELBO with use of gradient descent.

The modeling of $p_\phi(z_j|\mathbf{r}_j)$ is done similar as for the regular VAE algorithm and has the following structure:

$$p_\phi(z_j|\mathbf{r}_j) : \begin{cases} h_1 = ReLU(FC_{|\mathbf{r}_j| \to 128}(\mathbf{r}_j)) \\ h_2 = ReLU(FC_{128 \to 128}(h_1)) \\ \mu = FC_{128 \to 128}(h_2) \\ \sigma = exp(\frac{1}{2}FC_{128 \to 128}(h_2)) \end{cases} \tag{7.9}$$

This leads to the following distribution that is specified over the latent variables:

$$p_\phi(z_j|\mathbf{r}_j) \sim \mathcal{N}(\mu, \sigma\mathbf{I}) \tag{7.10}$$

Since a row $\mathbf{r}_j$ consists of $2N_c + N_d$ variables, the generator outputs a joint probability distribution over these variables. It is assumed that $\alpha_{i,j}$ follows a Gaussian distribution and all $\beta_{i,j}$ and $\mathbf{d}_{i,j}$ follow a categorical probability mass function. Formally, the structure of the generator network can be written as:

$$p_\theta(\mathbf{r}_j|z_j) : \begin{cases} h_1 = ReLU(FC_{128 \to 128}(z_j)) \\ h_2 = ReLU(FC_{128 \to 128}(h_1)) \\ \bar{\alpha}_{i,j} = tanh(FC_{128 \to 1}(h_2)), \quad 1 \le i \le N_c \\ \hat{\alpha}_{i,j} \sim \mathcal{N}(\bar{\alpha}_{i,j}, \delta_i), \quad 1 \le i \le N_c \\ \hat{\beta}_{i,j} \sim softmax(FC_{128 \to m_i}(h_2)), \quad 1 \le i \le N_c \\ \hat{\mathbf{d}}_{i,j} \sim softmax(FC_{128 \to |D_i|}(h_2)), \quad 1 \le i \le N_d \end{cases} \tag{7.11}$$

With $\hat{\alpha}_{i,j}$, $\hat{\beta}_{i,j}$ and $\hat{\mathbf{d}}_{i,j}$ being random variables and δ_i being a parameter that is leared during the train process (Xu et al., 2019). The joint probability distribution over the output of the generator can then be defined as:

$$p_\theta(\mathbf{r}_j|z_j) = \prod_{i=1}^{N_c} \mathbb{P}(\hat{\alpha}_{i,j} = \alpha_{i,j}) \prod_{i=1}^{N_c} \mathbb{P}(\hat{\beta}_{i,j} = \beta_{i,j}) \prod_{i=1}^{N_d} \mathbb{P}(\hat{\alpha}_{i,j} = \alpha_{i,j}) \tag{7.12}$$

7.3 Statistical Tests Statistics

In general, distinction can be made between one-sample tests and two-samples tests. A one-sample test is conducted to draw conclusions about a population based on a single sample from that distribution. Contrariwise, a two-sample test is used to draw conclusions about the relation between two populations, based on a sample from each of the populations.

7.3.1 Two-sample Kolmogorov-Smirnov Test Statistic

The two-sample Kolmogorov-Smirnov test (KS-test) is a non-parametric test, used to compare the cumulative distribution functions of two samples. More specifically, the KS test statistic (D) calculates the largest (vertical) distance between the empirical cumulative distribution functions of two samples and can be written as:

$$D_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)| \tag{7.13}$$

with $F_{1,n}(x)$ and $F_{1,m}$ being the cumulative distribution functions of sample 1 and 2 with n and m number of records, respectively. The sup function is the supremum function, which explicitly aims to finds the largest distance between the two cumulative distribution functions.

The two sample KS-test will not be used to reject a hypothesis about the relation between two samples. More specifically, the test will only be used to measure the distance between two cumulative distribution functions. This will be discussed more extensively in the subsection 8.2.1.

7.4 Correlation Measures

To investigate if a relationship exists between two features, their correlation can be calculated. However, it depends on the feature type what method should be used. Since the RTD includes continuous and categorical features, three possible feature combination must be distinguished.

First, to calculate the correlation between continuous features, Pearson'r correlation coefficient (K. Pearson, 1896) is used. Second, to calculate the correlation between two categorical features, Theil's U (Shannon, 1948) is used. Finally, for a feature combination consisting of a continuous and categorical feature, the correlation ratio (Fisher, 1970) is used.

In this reserach, except for the machine learning-based evaluation, all discrete features are assumed to be categorical (i.e., nominal). Therefore, the correlation between features, of which at least one is a categorical feature, will never be below 0. This holds since for a negative correlation, both features need to have an ordering. This means that for a feature combination with at least one discrete feature, this feature must be ordinal.

7.4.1 Pearson's Correlation Coefficient

Pearson's Correlation Coefficient, ρ, is used to measure if there exists a linear relationship between two continuous features X and Y. The coefficient is expressed as the covariance between two continuous features divided by the product of the standard deviations of both continuous features. Mathematically, this can be written as:

$$\rho = \frac{\sum_i^N (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i^N (x_i - \overline{x}) \sum_i^N (y_i - \overline{y})}} \tag{7.14}$$

ρ can have values in the interval $[-1, 1]$ with positive and negative values indicating a linear relationship with positive and negative slope.

7.4.2 Kendall's Tau Correlation Coefficient

In contrast to Pearon's ρ, Kendall's tau (Kendall, 1938) calculates the rank correlation coefficient since it measure ordinal correlation between two continuous features X and Y. More specifically, a rank correlation coefficient measures to what extent two rankings are similar.

Kendall's τ knows three variants: τ_a, τ_b and τ_c. In contrast to τ_a, τ_b makes adjustments for ties. In addition, τ_c is preferred over τ_b, only if the number of possible rankings differs for X and Y. Since only samples will be used with an equal amount of possible rankings and for which ties can appear, Kendall's τ_b will be used from now on.

Kendall's τ_b makes use of observation pairs of X and Y, $\{(x_i, y_i), (x_j, y_j)\}$, with $i \neq j$ and $i, j \in \{1, ..., N\}$. This means that there are a total of $\binom{N}{2} = \frac{N(N-1)}{2} = N_0$ possible pairs. Kendall's τ_b is calculated with the following formula:

$$\tau_b = \frac{C - D}{\sqrt{(N_0 - N_x) - (N_0 - N_y)}} \tag{7.15}$$

With C, D, N_x and N_y being the concordant pairs, discordant pairs, number of non - tied pairs in X and number of non - tied pairs in Y, respectively.
A pair $\{(x_i, y_i), (x_j, y_j)\}$ is

- tied in:
  - X, if $x_i = x_j$ or
  - Y, if $y_i = y_j$
    The total number of tied pairs in X and Y are N_x and N_y, respectively
- Concordant when:
  - $x_i > x_j$ and $y_i > y_j$ or
  - $x_i < x_j$ and $y_i < y_j$
- Discordant when:
  - $x_i > x_j$ and $y_i < y_j$ or
  - $x_i < x_j$ and $y_i > y_j$

Equivalent to Pearson's ρ, Kendall's τ_b can have values in the range $[-1, 1]$ with positive values indicating a positive relation between the ordering of X and Y and vice versa. Table **??** can be used to interpret the strength of the correlation coefficient.

7.4.3 Theil's U

Theil's U, also known as the "Uncertainty Coefficient" (Shannon, 1948), can be used to determine the degree of association between two nominal variables. This correlation metric is based on conditional entropy between the nominal features. More specifically, for two nominal variables X and Y, the conditional entropy is described as the amount of information *required* to obtain the outcome of Y, given a known value for X. Mathematically, the conditional entropy for Y, given X can be written as:

$$H(Y|X) = -\sum_{x,y} P_{X,Y}(x, y)\, log\, P_{Y|X}(y|x) \tag{7.16}$$

With $P_{X,Y}(x, y)$ and $P_{Y|X}(y|x)$ being the joint distribution over X and Y and the conditional distribution of Y, given X, respectively. In addition, the (total) entropy for a single variable Y, can be mathematically written as:

$$H(Y) = -\sum_{y} P_Y(y)\, log\, P_Y(y) \tag{7.17}$$

In this case, $H(Y)$ should be thought of as the total information necessary to explain the outcomes of Y. Then, combining the conditional entropy and single total entropy for Y, the fraction of Y that can be explained given X is written as:

$$U(Y|X) = \frac{H(Y) - H(Y|X)}{H(Y)} \tag{7.18}$$

The values of U are in range $[0, 1]$. Values closer to 1 indicate strong correlation and vice versa. This makes sense, since when there is *is no information required* to explain the outcomes of Y, given X (i.e., $H(Y|X) = 0$), this means that Y and X are identical and therefore fully correlated.

7.4.4 Correlation Ratio

Karl Pearson introduced the correlation η ratio as a part of Analysis of Variance (ANOVA) and is used to calculate the correlation between a categorical and a continuous feature. Having two samples, the correlation ratio measures the relation between the statistical dispersion (i.e., how stretched or squeezed a distribution is, measured in terms of standard deviation) of the continuous values within a particular category and the dispersion in the whole population. Let X and Y be a categorical and a continuous feature, respectively, with n_x being the number of continuous values in category x, n being the total number of continuous values and

$$\overline{y}_x = \frac{\sum_{i=1}^{n_x} y_{x,i}}{n_x} \quad \text{and} \quad \overline{y} = \frac{\sum_x n_x \overline{y}_x}{n} \tag{7.19}$$

be the mean of the values in category x and the overall mean of the continuous values, respectively. The correlation ratio is described as the weighted variance of the category means divided by the total variance. Mathematically this can be written as:

$$\eta^2 = \frac{\sigma_{\overline{y}}^2}{\sigma_y^2}\text{ , with}\quad \sigma_{\overline{y}}^2 = \frac{\sum_x n_x(\overline{y}_x - \overline{y})^2}{\sum_x n_x} \text{ and } \sigma_y^2 = \frac{\sum_{x,i}(y_{xi} - \overline{y})^2}{n} \tag{7.20}$$

The values of η are in the range $(0, 1)$, with the limits $\eta = 0$ and $\eta = 1$ representing no dispersion among the means of different categories and no dispersion within the particular categories, respectively. However, high values within the range indicate strong correlation and vice versa.

7.5 Performance Metrics

In supervised machine learning, the predictive performance of a model is often quantified by performance metrics such as the Coefficient of Determination (R^2), the Mean Absolute Error (MAE) and Mean Absolute Percentage error (MAPE). Thereby, MAE and MAPE can also be used as prediction error metrics between neural network output values and known target values (Amin-Naseri & Soroush, 2008).

7.5.1 Coefficient of Determination

The coefficient of determination, R^2, describes to what extent a model is able to explain the variability of a target feature with respect to a baseline model, the grand mean ($\overline{y}$). The variability explained by the baseline model is calculated with the formula $\sum_{i=1}^{N}(y_i - \overline{y})^2/N$. The percentage of variability that is *not* explained by the model is calculated with the formula $\sum_{i=1}^{N}(y_i - \hat{y_i})^2/N$. Using both equations, the variability of the target feature described by the model can be written as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y_i})^2}{\sum_{i=1}^{N}(y_i - \overline{y})^2} \tag{7.21}$$

The values of R^2 are in the range $(-\infty, 1]$. When the model predicts the values of the target feature perfectly, the distance between the predicted value and the observed value is 0. This results in (the nominator of) the fraction in the formula being equal to 0, and therefore $R^2 = 1$. However, in case the model predicts values that have a larger distance to the observed value than the grand mean, this leads to a negative value for R^2.

7.5.2 Mean Absolute Error

The Mean Absolute Error (MAE) quantifies the mean absolute difference between observed and predicted values of a target feature. The MAE is calculated with the following formula:

$$MAE = \frac{\sum_{i=1}^{N}|y_i - \hat{y_i}|}{N} \tag{7.22}$$

Since the absolute difference is taken between observed and predicted values, it holds that MAE $\in \mathbb{R}^+$. Finally, because the MAE is an error metric, it is desired for prediction models to have a low MAE, meaning close to 0.

7.5.3 Mean Absolute Percentage Error

The Mean Absolute Percentage Error (MAPE) differs slightly from MAE since it calculates the mean absolute distance between predicted and observed values as a percentage of the observed values. The MAPE is calculated with the following formula:

$$MAPE = \sum_{i=1}^{N} |\frac{y_i - \hat{y}_i}{y_i}| * \frac{100}{N} \tag{7.23}$$

Since the absolute percentage of difference between observed and predicted values is taken, it holds that MAPE $\in \mathbb{R}^+$. Finally, because the MAPE is an error metric, it is desired for prediction models to have a low MAPE, meaning close to 0.

7.5.4 Total Absolute Percentage of Error

This metric is a variation of the MAPE, designed for this research specifically. This means that the TAPE does not originate from scientific literature, or at least not under this name. The Total Absolute Percentage of Error, for a categorical feature, quantifies the total absolute difference between predicted ($\hat{Y}$) and observed (Y) count data as a percentage of total records. Mathematically this can be written as:

$$TAPE = \frac{\sum_{i=1}^{C} |y_i - \hat{y}_i|}{2} * \frac{100}{N} \tag{7.24}$$

With C, N, y_i and $\hat{y}_i$ being the number of unique categories in the categorical feature, the total number of records, the number of observed records for category i and the number of predicted records for category i, respectively. Assume C_0 to be the number of categories for which the predicted count data is equal to 0.

The total absolute difference must be diveded by 2 since the error is calculated double when calculating the absolute difference between each category. Values for the TAPE are in the range $[0, 1)$. The TFE cannot be equal to 1, since the sum of the predicted count data and observed data must be the same. Therefore, at least $C - C_0$ records will be assigned to a category correctly. It is desired for categorical features to have a TAPE equal to 0, since this indicates that no predicted records where assigned to a category incorrectly.

7.6 Regression Models

Regression is a method to determine the relationship between a single dependent feature and (multiple) independent features. Several regression algorithms, such as linear regression, support vector regression and neural networks, work well with continuous data. However, in many real world applications, the data is not always solely continuous. Tree-based algorithms are well suited to perform regression with mixed type dependent variables (Kim & Hong, 2017).

7.6.1 Light Gradient Boosting Machine

The Light Gradient Boosting Machine (LightGBM), developed by Microsoft, is a tree-based regression algorithm and works well with high dimensional data. In addition, the LightGBM is insensitive to noise and can handle missing values in the independent features (Li et al., 2018).

Clappform's ERV model is based on the light gradient boosting machine algorithm. Due to confidentiality, the hyperparameter values will not be disclosed. The train data of the model consists of three elements. These elements are the RTD data, location data and livability data. Again, due to confidentiality, the content of the location data and livability data will not be described. However, the three elements all include the features "neighborhood", "municipality" and "postal code". Since the latter feature is removed from the RTD, the three elements are merged on the features "neighborhood" and "municipality" and will together form the train data.

Experimental Set-up

In this chapter, the complete process regarding learning real estate transaction data is described and is divided into two sections. In the first subsection, 8.1, the used deep generative models and their configurations are described. Hereafter, in section 8.2, the statistical analysis and machine learning efficacy analysis, which are used to assess the quality of synthetic data, are discussed in subsections 8.2.1 and 8.2.2, respectively.

8.1 Modeling Real Estate Transaction Data

This section reviews how deep generative models are used to learn real estate rental data. First, in subsection 8.1.1, it is described how a test sample and 12 train samples are extracted from the total data set. This is done in order to train deep generative models on multiple (increasing) sample sizes to analyze to what extent the synthetic data quality depends on the input size of the deep generative models. Hereafter, in subsection 8.1.2, the characteristics and the hyperparameter settings of the used models are discussed.

8.1.1 Train size

A test set is required to measure the machine learning efficacy regarding the ERV model (Clappform's real estate rental price prediction model). Therefore, the RTD is split into a (mutually exclusive) train and test set. The splitting is done with the use of the "train_test_split" function from the Python package "sklearn". This function contains the parameter "test_size", indicating what percentage of the RTD must be put into the test set. The test set is randomly sampled from the total data set and a value of 20% is set for this value. This results in a train and test set of 68.952 and 17.238 records, respectively.

The remaining train set can be used to train deep generative models. In total, 12 samples are extracted from the train set with 350, 700, 1.050, 1.500, 2.000, 3.000, 5.000, 7.500, 10.000, 20.000, 40.000 and 68.952 records, respectively. First, 350 records are sampled and removed from the total train set to obtain the sample of size 350. Hereafter, an additional 350 records are sampled and removed from the remaining train set and added to the sample of size 350 to obtain the train sample of size 700, and so on. This means that the train samples are nested, meaning that train sample of size 350 is included in 700, 700 in 1.050, etc. This is done to make sure that the deep generative models that are trained on large train samples are at least also trained on the train samples that were used for models that are trained on small samples. The sampling process is done in a stratified fashion on the feature "municipality". Stratified sampling ensures that the distribution of the feature on which is stratified is retained in each sample. More specifically, the sampling process ensures that records for each unique category in "municipality" are present in the sample and are similarly distributed as in the total train set. Since there are 349 unique categories for "municipality" in the total train set, the size of the smallest train sample is set to 350 records. In addition, when sampling more than 350 records, the number of records per category is sampled proportionally to the distribution of the categories in the total train set.

The feature "municipality" is chosen to stratify on to proportionally distribute real estate rental transactions of all municipalities in the total train set in each sample. This is done since it is important for Clappform to know how well deep generative models can learn RTD throughout all present municipalities in the data. If records were sampled randomly (in a non-stratified fashion), small samples would not contain records from feature categories that represent a strong minority, with high probability.

8.1.2 Deep Generative Models

The used deep generative models, CTGAN and TVAE, are both trained on all train samples, resulting in submodels

- $CTGAN_1, ..., CTGAN_{12}$ and
- $TVAE_1, ..., TVAE_{12}$

Where the subscripts indicate the sample size on which a submodel is trained. Subscript 1 represents the smallest sample (350 records), up to subscript 12 which represents the total train set (68.952 records).

For both models, general characteristics are initialized and kept constant for each train sample size. This means that all submodels are bounded and are subject to constraints. When a submodel is bounded, it will learn the upper and lower bound values for all features in the train sample and is only able to generate synthetic data that is within these bounds. Similarly for constraints, all submodels learn these constraints and will only generate synthetic data that satisfies the constraints. The used constraints are:

- Unique combinations between features ”neighborhood” and ”municipality” must be preserved
- Unique combinations between features ”parking type” and ”parking availability” must be preserved
- In case a value is listed for the feature ”number of floors building”, ”floor level” can’t exceed.
- In case a value is listed for the feature ”total rooms”, ”total bedrooms” can’t exceed.

However, the configuration of hyperparameters is different for both models. Deep generative models are notorious for their long run time when taking big data as input. In addition, having several (and unbounded) parameters can lead to an enormous hyperparameter space representing all possible combinations. Therefore, due to a lack of time and GPU compute power, it is unfeasible to explore this parameter space to find the optimal configuration for all 24 submodels. A single configuration per model will be used. For both models, their configurations will be the default configuration that is used in the SDV package, leading to the following hyperparameter configurations:

CTGAN Hyperparameter Configuration

Epochs: 300, batch size: 500, log frequency: True, latent space dimension: 128, generator dimensions: $(256, 256)$, discriminator dimensions: $(256, 256)$, generator learning rate: $2e^{-4}$, generator decay: $1e^{-6}$, discriminator learning rate: $2e^{-4}$, discriminator decay: $1e^{-6}$, discriminator steps: 1.

TVAE Hyperparameter Configuration

Epochs: 300, batch size: 500, log frequency: True, latent space dimension: 128, encoder dimensions: $(128, 128)$, decoder dimensions: $(128, 128)$, regularization term: $1e^{-5}$, loss factor: 2

After training, a submodel can be used to generate synthetic data. How well a submodel was able to learn the probability distribution of the RTD is analyzed by means of a quality analysis[1] on the synthetic data. This will be explained more extensively in section 8.2.

8.2 Quality Analysis

The quality analysis of synthetic data consists of a statistical analysis and a machine learning efficacy analysis, per submodel. First, the statistical analysis is discussed in subsection 8.2.1. Hereafter, in subsection 8.2.2, the machine learning efficacy analysis is discussed.

8.2.1 Statistical Analysis

In the statistical analysis, it is investigated how well a submodel learned the empirical probability distribution of the RTD by quantifying the deviation between statistical properties of empirical samples and synthetic samples. For each submodel, five deviation metrics are calculated 50 times between an empirical sample, that is sampled in a stratified fashion on the feature ”municipality”, and a synthetic sample. For each particular submodel, the 50 empirical and synthetic samples have an equal size to the train sample of the submodel.

Hereafter, the similarity between two samples is quantified with the use of the similarity score. This score is obtained by taking the mean of five individual similarity metrics. These similarity metrics are calculated with the use of the deviation metrics. Since the deviation metrics are calculated 50 times, the total similarity score is also calculated 50 times. In this way, a quantification of the variability level of the similarity score is obtained per submodel.

[1]What is important to note, is that each submodel is trained only once on an empirical train sample. Therefore, the quality analysis is to some extent dependent on the used train samples. This limitation will be discussed more extensively in the discussion.

The similarity metrics are based on the following deviation metrics:

1. MAPE between basic statistics for continuous features.
2. Kolmogorov-Smirnov test statistic for continuous features.
3. TAPE between count data for categorical features.
4. Kendalls Tau rank correlation coefficient between correlation matrices.
5. MAE between correlation matrices.

The similarity metrics are shown at the end of this section.

MAPE between basic statistics for continuous features

The MAPE is often used because it is an intuitive way of interpreting relative errors. In addition, the MAPE is frequently used in real-world applications if the predicted value is way above 0 (De Myttenaere et al., 2016). Basic statistics for continuous features are the sample mean, median and variance and are known to be significantly larger than 0 for continuous features in the RTD. The smaller the MAPE, the better a submodel was able to reproduce the basic statistics of the empirical sample in the synthetic sample.

Kolmogorov-Smirnov test statistic for continuous features

The (two-sample) Kolmogorov-Smirnov test statistic (D) quantifies the largest distance between two cumulative distribution functions. Therefore, this statistic could be used to quantify the distance between the cumulative distribution functions of an empirical sample and a synthetic sample. Although the MAPE for basic statistics could be low, there might still be dissimilarities between an empirical and a synthetic sample, which are not captured by this metric. The lower D is, the closer the cumulative distribution function of the empirical and synthetic samples are. Therefore, D is a complementary metric to the MAPE for basic statistics to quantify the similarity between an empirical and synthetic continuous sample.

TAPE between count data for categorical features

The count data for a categorical feature sample includes the number of observed records per unique category. In contrast with the basic statistics for continuous features, predicted count data (i.e., count data for a synthetic sample) can become very low or even equal to 0 for multiple categories. Using MAPE as a distance metric could lead to a disproportional error value. For example, when a categorical feature with four unique categories in the empirical sample has count data $[12, 1, 1, 1]$ and the predicted count data is $[15, 0, 0, 0]$, this leads to a MAPE of 80% while only 3/15 of the records were assigned to a category incorrectly.

Therefore, the TAPE[2] is used to calculate the error between empirical count data and predicted count data. This metric measures the absolute number of records that are incorrectly assigned to a category as a percentage of the total empirical records. For the same example as just mentioned, the TAPE would result in $\frac{6}{2*15} * 100 = \frac{3}{15} * 100 = 20\%$. This value is much more in proportion with respect to the error calculated with MAPE. The lower the TAPE is, the better a submodel was able to reproduce the empirical count data in the synthetic sample.

Kendalls Tau rank correlation coefficient between correlation matrices

First, the correlation matrices are obtained for all feature combinations in the empirical and synthetic sample, respectively. Hereafter, they are converted into one-dimensional vectors after which the correlation between these vectors is calculated. In (Brenninkmeijer & Hille, 2019), the correlation between the correlation vectors is calculated with Pearson's r. This method only determines if there exists a linear relationship between two univariate vectors, but not what the slope of the linear relation is. Since it is desired that the vectors are similar, the desired slope of the linear relationship is 1. Since Pearson's r could return a high correlation value for a linear relation with a slope different than 1, using this method could result in misleading values about similarity of the two vectors (and thus correlation matrices). Therefore, instead of Pearson's r, Kendalls τ is used. This is a rank based correlation coefficient which measures to what extent the rankings of both vectors are similar. The higher Kendalls τ, the better a submodel was able to reproduce the correlations of the empirical features into the synthetic sample.

[2]This metric is designed for this research specifically. That means that it could be found in scientific literature under a different name and for different purposes.

MAE between empirical and synthetic correlation matrices

Since all correlation values are all between 0 and 1, the MAE is used instead of the MAPE to quantify the error between the two correlation matrices. The smaller the MAE between the empirical correlation matrix and a synthetic correlation matrix, the better a submodel was able to reproduce the correlations from the empirical sample into the synthetic sample.

Thereby, a low Kendalls τ value does not imply a low MAE and vice versa. This is because when Kendall's τ is relatively low, the rankings of the correlation vectors is not similar and thus the MAE calculates errors between correlation values of non-corresponding feature combinations. Therefore, it is desired to have a high value for both Kendalls τ and MAE simultaneously.

Statistical similarity score

To quantify the overall statistical score between an empirical and a synthetic sample, the mean is taken over five similarity metrics. These similarity metrics are calculated with use of the five before mentioned deviation metrics and are the following:

1. $1 - \frac{MAPE}{100}$
2. $1 - D$
3. $1 - \frac{TAPE}{100}$
4. Kendalls τ
5. $1 - MAE$

For deviation metrics MAPE, D, TAPE and MAE, a high value indicates low statistical similarity. In addition, the MAPE and MAE are metrics that can surpass the value of 1. Therefore, for similarity metrics $1, 2, 3$ & 5, the lower bound is set to 0. Values close to 1 represent high statistical similarity and vice versa. For similarity metric 4, only correlation coefficients in the range $[0, 1]$ are possible with values close to 1 indicating high similarity. Therefore, this metric is kept as it is. Finally, the similarity score for a by a submodel generated synthetic sample, S, is calculated by taking the mean over similarity metrics $1 - 5$. Again, since all deviation metrics (and thus similarity metrics) are calculated 50 times per submodel, the similarity score is also calculted 50 times. In this way, also a quantification of the variability level of the similarity score can be obtained per submodel.

The similarity score indicates how well a submodel was able to reproduce statistical properties from the RTD into synthetic samples and lies in the range $[0, 1]$, where 0 and 1 indicate no statistical similarity and perfect statistical similarity, respectively.

8.2.2 Machine Learning Efficacy Analysis

The machine learning efficacy analysis consists of two parts. In the first part, the performance of the ERV model is analyzed after replacing empirical train samples with synthetic train samples. Hereafter, the performance is analyzed after augmenting the empirical train samples to balance an imbalanced feature.

Replacing empirical train samples

First, the performance of the ERV model is calculated after being trained on empirical samples. This is done for each train sample size and will be called the basic ERV performances. To obtain a variability level of the basic ERV performances, the calculation is done 50 times per train sample size. The empirical samples are sampled from the total train data in a stratified fashion on the feature "municipality". Thereby, performances of the basic ERV model are calculated in terms of R^2 and will be used as benchmark in the machine learning efficacy analysis.

In the first part of the analysis, the empirical train samples are replaced by synthetic samples, whereafter the performance of the ERV model is calculated on the total test set[3]. For both models, this is done 50 times per

[3]When training the ERV model on synthetic data, the hyperparameters are not again fitted to the data. Instead, the hyperparameters of the original ERV model are used. The original model is developed by the data analytics team of Clappform. The performances of the models that are trained on synthetic data might therefore be suboptimal. This limitation will be elaborated on more extensively in the discussion.

train sample size to obtain the performance level variability. Finally, for both models the performances are compared to the performances of the basic ERV model per train sample size.

Augmenting empirical train samples

In the second part of the analysis, the ERV performance is analyzed after augmenting the empirical train samples with synthetic records to balance an imbalanced categorical feature. A regression algorithm tends to favor the majority class of a feature when learning from a sample in which the amount of training data for a minority category is substantially smaller than the training data for a majority category (Nguyen et al., 2008). In the RTD, the categorical feature ”interior type” is imbalanced. This feature has 3 unique categories: ”shell”, ”furnished” and ”upholstered”.

First, for the unique categories ”shell”, ”furnished” and ”upholstered”, the percentage of records per category is calculated. This is done 50 times done for each train sample size to obtain the variability level of the percentage of records per category. Again, a sample is sampled from the total train set in a stratified fashion on the feature ”municipality”. Table 8.1 shows the results.

train sample size	**350**	**700**	**1.050**	**1.500**	**2.000**	**3.000**	**5.000**	**7.500**	**10.000**	**20.000**	**40.000**	**68.952**
% shell	5.85 ± 1.21	6.14 ± 0.97	6.14 ± 0.59	6.25 ± 0.61	6.26 ± 0.54	6.23 ± 0.44	6.23 ± 0.27	6.26 ± 0.27	6.28 ± 0.21	6.29 ± 0.14	6.27 ± 0.083	6.26 ± 0.00
% furnished	27.69 ± 1.96	27.40 ± 1.48	27.49 ± 1.43	27.17 ± 1.06	27.19 ± 0.81	27.22 ± 0.72	27.33 ± 0.56	27.45 ± 0.43	27.28 ± 0.36	27.30 ± 0.24	27.31 ± 0.15	27.29 ± 0.00
% upholstered	55.15 ± 2.34	54.88 ± 1.77	55.16 ± 1.49	55.13 ± 1.11	54.78 ± 0.93	55.03 ± 0.79	54.86 ± 0.61	54.93 ± 0.41	54.93 ± 0.34	54.96 ± 0.28	54.98 ± 0.17	54.99 ± 0.00

Table 8.1: For each unique category, the mean and standard deviation of the percentage of records of 50 samples is calculated for each sample size. Note that for each train sample size the sum of the percentages per category is not equal to 100%. This is because for the feature ”interior type” contains missing values.

In table 8.1, it is shown that, for each train sample size, the category ”shell” represents a strong minority percentage of the total number of records. Therefore, it will be researched if augmenting empirical train samples with synthetic shell records has impact on the performance of the ERV model. Again, this research can be split into two parts.

First, the performance of the basic ERV model will be analysed on mutually exclusive test sets that only contains shell, furnished or upholstered records, respectively. The shell test set is obtained by removing all records from the total test set that do not represent an interior type equal to shell. The same method is applied to obtain test sets that only contain furnished and upholstered records. The resulting test sets for shell, furnished and upholstered contain 998, 4827 and 9402 records, respectively. The performance of the basic ERV model on all three test sets is shown in figure 8.1.

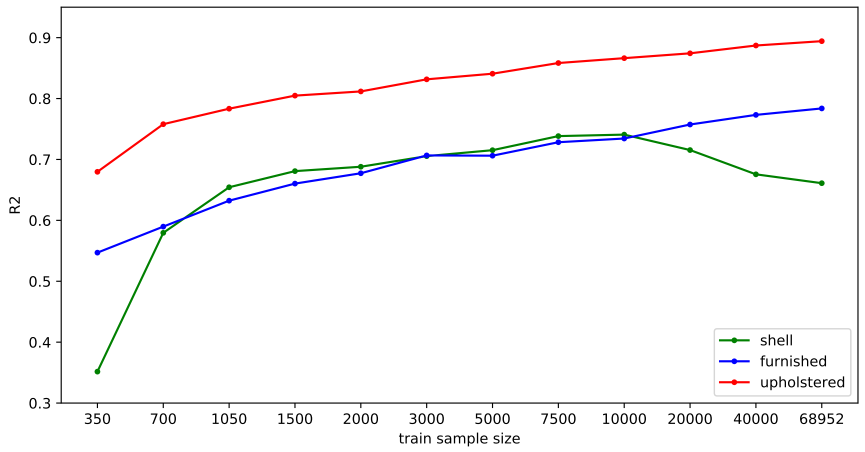


Figure 8.1: Mean performances of basic ERV model on 50 empirical samples per sample size on shell (green), furnished (blue) and upholstered (red) test sets. The standard deviations of the performances are not shown in this figure to keep the plot as clear as possible. To see the mean performances of the basic ERV model including standard deviations, see figures A.1, A.2 and A.3 in subsection A.1.1 in the appendix.

In figure 8.1, a decrease in performance of the basic ERV model on the shell test set appears for train sample sizes larger than 10.000. This is despite the fact that the percentage of records for each category stays more or less constant (see table 8.1). However, since the sample size increases, the absolute difference between number of shell records and number of furnished and upholstered records becomes larger. The decrease of performance might be due to the shell records getting overruled by a relatively larger amount of furnished and upholstered records in train samples larger than 10.000 records. Therefore, synthetic shell records are added to the empirical data samples larger than 10.000 to research if this has improves the performance of the basic ERV model on the shell test set. First 10.000 shell records are added, hereafter 20.000 records are added. In addition, the performance of the basic ERV model on furnished and upholstered test sets will be analysed to see if these performances are not negatively influenced by augmenting the empirical train data. The performances on all three test sets are done 50 times to obtain the performance level of the ERV model trained on augmented samples.

Second, the performance of the ERV, after augmenting the empirical train samples with synthetic shell records, is analysed on the complete test set. This is done since Clappform is also interested in the performance of the ERV model on the total test set instead of only the performance on separate test sets for shell, furnished and upholstered. However, an improved performance of a particular test set such as the shell test set might indicate that balancing an imbalance features could improve the performance of the ERV model on the total test set. The performance of the ERV model on the total test set, trained on augmented empirical train samples, is compared with the performance of the basic ERV model. Again, the performance of the augmented ERV is calculated 50 times to obtain the performance variablility level.

Results

In this section, the results of the statistical analysis and machine learning efficacy analysis are addressed. First, in section 9.1, the results of the statistical analysis are discussed. This is done separately for TVAE and CTGAN submodels in subsections 9.1.1 and 9.1.2, respectively. Hereafter, the results of the statistically best performing models are discussed on a more detailed level in subsection 9.1.3. Second, in section 9.2, the results for the machine learning efficacy analysis are discussed. This analysis is divided into two parts. The first part includes the results about the ERV model performance after replacing the empirical train samples with synthetic samples. Hereafter, in the second part, the results of the ERV model performance are discussed after augmenting the empirical train samples with synthetic records to balance the imbalanced categorical feature "interior type".

9.1 Statistical Analysis

In this section, the similarity score for all submodels is discussed. For each submodel, the similarity score is calculated by taking the mean of five statistical similarity metrics. These metrics are based on the deviation between statistical properties of an empirical sample and an, by a submodel generated, equal size synthetic sample, and are described in subsection 8.2.1. The final similarity score per submodel is calculated by taking the mean of 50 similarity score observations, which are calculated for 50 empirical and synthetic samples[1], to obtain the variability level of the similarity score. For clarity, the mean similarity score is from now called the similarity score. First, the similarity scores for TVAE submodels are discussed in subsection 9.1.1, whereafter the similarity scores for CTGAN submodels are discussed in subsection 9.1.2. Hereafter, for the models with the highest similarity score, the results of the five similarity metrics are discussed in subsection 9.1.3.

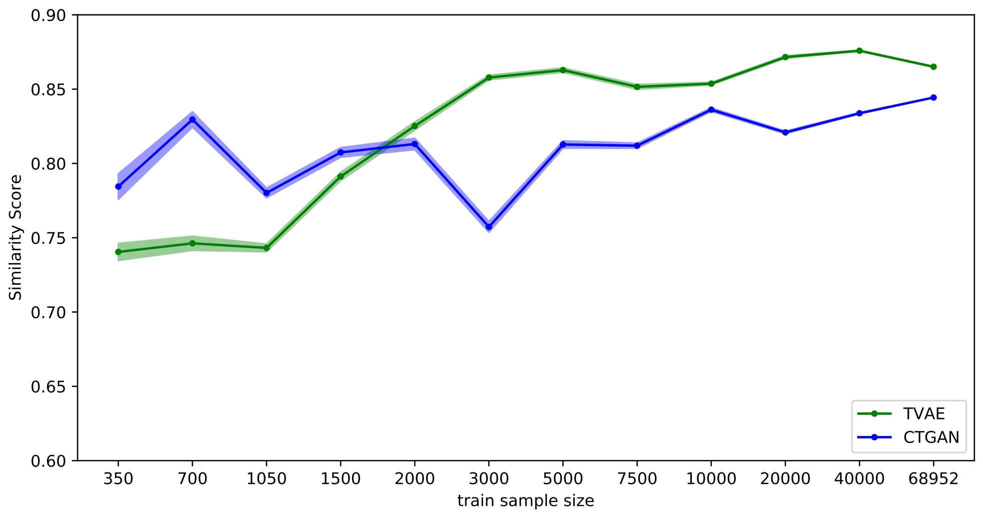


Figure 9.1: Similarity score with standard deviation for all TVAE submodels (green line) and CTGAN submodels (blue line). Recall that submodel 1 is trained on sample size 350, submodel 2 on train sample size 700, up to submodel 12 on the full train sample of size 68.952.

9.1.1 TVAE Similarity Score

In figure 9.1, the green line represents the similarity score with standard deviation for all TVAE submodels. The similarity scores for submodels 2, 3 and 9 are approximately constant with respect to their predecessor submodels. Thereby, the similarity scores even slightly decreases for submodels 3, 8 and 12. TVAE submodel 11 performs best with a similarity score of 0.88. The submodel that performed the worst is submodel 1 with a

[1]All 50 empirical and synthetic samples are independently sampled from the total empirical train set (in a stratified fashion on the feature "municipality") and from a submodel, respectively.

similarity score of 0.74.

The green line clearly shows an increase in similarity score when train samples are enlarged, indicating that TVAE submodels on average obtain a larger similarity score when being trained on a larger sample.

9.1.2 CTGAN Similarity Score

The blue line in figure 9.1 represents the similarity score with standard deviation for all CTGAN submodels. For submodels 1-7, the blue line shows more variability in similarity score between consecutive models compared to the green line. The similiarty scores for these models fluctuates between a maximum of 0.83 for submodel 1 and 0.75 for submodel 6. Submodels 8-12 show less variability in similarity score between consecutive models. The best and worst performing CTGAN submodels are submodel 12 and 6 with a similarity score of 0.84 and 0.75, respectively.

For the blue line, an increasing trend is also visible, starting at submodel 6. This indicates that the minimum required sample size for the CTGAN model, to generate synthetic data with a more stable similarity score, is larger than for the TVAE model. Thereby, it can be concluded that CTGAN models trained on a sample size larger than or equal to 7.500 on average obtain a larger similarity score.

9.1.3 Similarity Score Metrics

Table 9.1 shows the similarity metric values for the best performing submodels, which are TVAE submodel 11 and CTGAN submodel 12, respectively. The CTGAN submodel outperforms the TVAE with similarity metrics 1 and 2 on all continuous features. However, the TVAE model outperforms the CTGAN model with similarity metric 3 on 11/16 categorical features. Thereby, the TVAE submodel outperforms the CTGAN model with both similarity metrics 4 and 5.

Feature (combinations)	Similarity Metric	TVAE submodel 11	CTGAN submodel 12	Difference
listing price	1	0.92 ± 0.0037	$\mathbf{0.96 \pm 0.0026}$	0.04
listing size	1	0.91 ± 0.0030	$\mathbf{0.97 \pm 0.0021}$	0.06
listing price	2	0.93 ± 0.0026	$\mathbf{0.96 \pm 0.0015}$	0.03
listing size	2	0.95 ± 0.0028	$\mathbf{0.97 \pm 0.0012}$	0.02
parking availability	3	0.72 ± 0.0019	$\mathbf{0.86 \pm 0.00081}$	0.14
garden	3	0.95 ± 0.0024	$\mathbf{0.95 \pm 0.0014}$	0.00
storage	3	$\mathbf{0.97 \pm 0.0018}$	0.90 ± 0.0012	0.07
garage	3	$\mathbf{0.98 \pm 0.0023}$	0.92 ± 0.00089	0.06
balcony	3	$\mathbf{0.99 \pm 0.0016}$	0.92 ± 0.0016	0.07
floor level	3	$\mathbf{0.99 \pm 0.0011}$	0.87 ± 0.0014	0.12
number of floors building	3	$\mathbf{0.98 \pm 0.0012}$	0.84 ± 0.0016	0.15
total rooms	3	0.90 ± 0.0023	$\mathbf{0.94 \pm 0.0013}$	0.04
total bedrooms	3	0.90 ± 0.0029	$\mathbf{0.93 \pm 0.0019}$	0.03
maintenance status	3	$\mathbf{0.98 \pm 0.00046}$	0.85 ± 0.0010	0.13
energy label	3	$\mathbf{0.93 \pm 0.00078}$	0.86 ± 0.0011	0.07
residential type	3	$\mathbf{0.94 \pm 0.0019}$	0.91 ± 0.00074	0.03
interior type	3	0.90 ± 0.0023	$\mathbf{0.92 \pm 0.0014}$	0.02
parking type	3	$\mathbf{0.97 \pm 0.00058}$	0.86 ± 0.0013	0.11
neighborhood	3	$\mathbf{0.53 \pm 0.0021}$	0.29 ± 0.0012	0.24
municipality	3	$\mathbf{0.82 \pm 0.0022}$	0.56 ± 0.0013	0.26
all feature combinations	4	$\mathbf{0.71 \pm 0.0028}$	0.59 ± 0.0023	0.12
all feature combinations	5	$\mathbf{0.95 \pm 0.00027}$	0.93 ± 0.00020	0.02

Table 9.1: Similarity metrics for TVAE submodel 11 and CTGAN submodel 12. Since the similarity score is calculated on 50 empirical and synthetic samples, this means that the similarity metrics are also calculated 50 times. The mean similarity metrics, with standard deviation, are shown for each feature.

The largest performance differences are visible for features "neighborhood" and "municipality" with a difference of 0.24 and 0.26, respectively. Hereafter, "number of floors building" has the largest performance difference. Since the cardinalities of "neighborhood" and "municipality" are too large to show in a plot, a normalized count plot is shown for "number of floors building". Figures 9.2 and 9.3 show two observation of the normalized count data in an empirical train sample (blue bars) and a synthetic sample (orange bars). The orange bars in figure 9.2 and 9.3 show the normalized count data for a synthetic sample generated by TVAE submodel 11 and CTGAN submodel 12, respectively.

The TVAE submodel outperformed the CTGAN submodel in approximating the count data of the empirical train sample. This mainly becomes clear due to the large difference for the category missing values in figure 9.3. Thereby, the CTGAN submodel produced significantly too many records for category 11 in the synthetic sample.

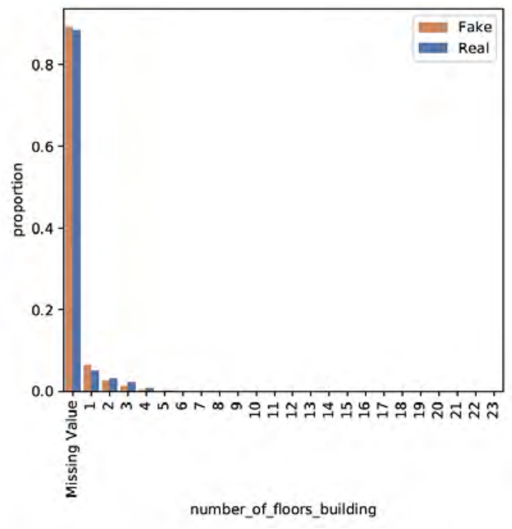


Figure 9.2: One observation of normalized count data of the feature "number of floors building" in both a synthetic and empirical sample. The orange bars indicate the count data of the synthetic sample produced by TVAE submodel 11. The blue bars indicate the normalized count data for the empirical train sample.

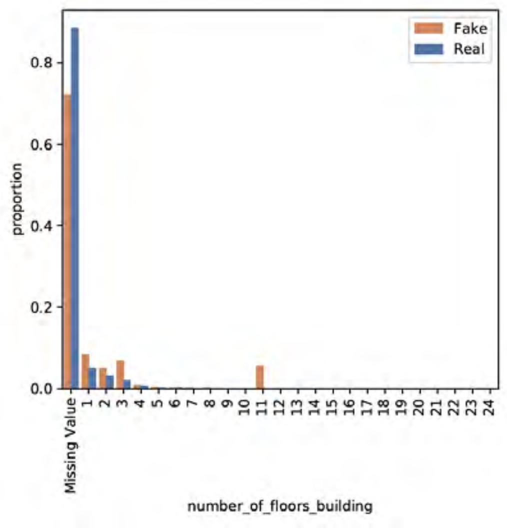


Figure 9.3: One observation of normalized count data of the feature "number of floors building" in both a synthetic and empirical sample. The orange bars indicate the count data of the synthetic sample produced by CTGAN submodel 12. The blue bars indicate the normalized count data for the empirical train sample.

For similarity metric 4, the CTGAN model performed 0.12 lower than the TVAE model. Figure 9.4 shows the correlation values between all feature combinations in the empirical train sample, synthetic data generated by TVAE submodel 11, and the difference between these two matrices.

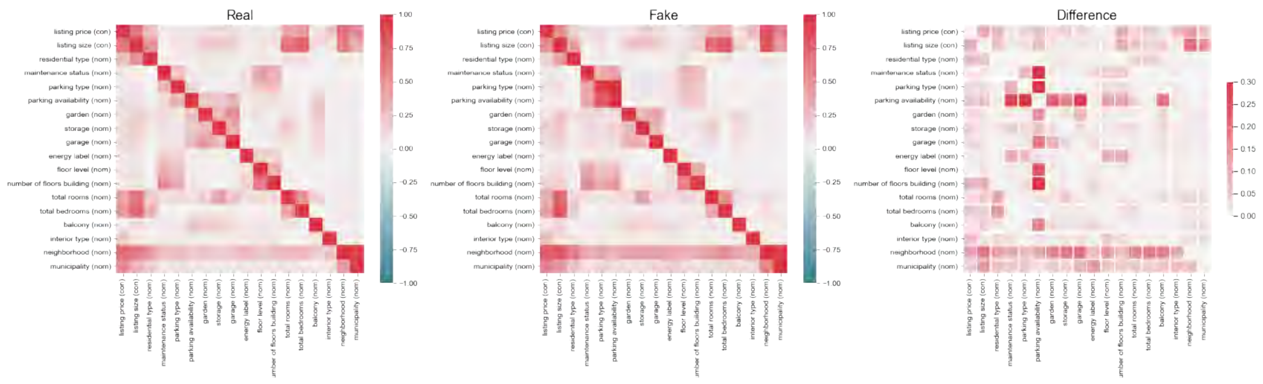


Figure 9.4: The correlation matrices for the empirical sample on which the TVAE submodel is trained (left), synthetic sample generated by the TVAE submodel (middle) and the absolute differences between these matrices (right). A dark red color indicates a high correlation value in the left and middle matrices and a high absolute difference in the right matrix.

In figure 9.4 it becomes clear that feature "parking availability" is correlated more strongly with "maintenance status" and "parking type" in the synthetic sample than in the empirical sample. Contrariwise, "parking availability" is less correlated with features "garden", "storage" and "garage" in the synthetic sample than in the empirical sample. Thereby, the feature "neighborhood" is more strongly correlated with all other features in the synthetic sample than in the empirical sample, except for "municipality"

Figure 9.5 shows that in the synthetic sample, the feature "parking availability" again has clearly different correlation values than in the empirical sample. Thereby, the feature "listing price" is significantly less correlated with "neighborhood" and "municipality".

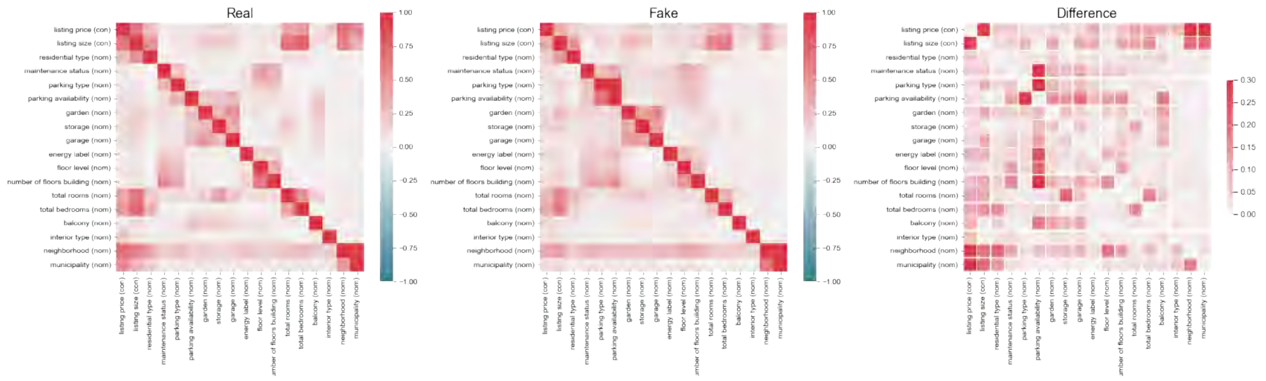


Figure 9.5: The correlation matrices for the empirical sample on which the CTGAN submodel is trained (left), synthetic sample generated by the CTGAN submodel (middle) and the absolute differences between these matrices (right). A dark red color indicates a high correlation value in the left and middle matrices and a high absolute difference in the right matrix.

Overall, the coordinates in the difference matrix in figure 9.5 are darker red colored than the values in the difference matrix in figure 9.4, indicating that the CTGAN submodel generated a synthetic sample in which correlations differ more from the train sample than for the TVAE submodel.

9.2 Machine Learning Efficacy Analysis

In this section, the results for both parts of the machine learning efficacy analysis are addressed. The first part, described in subsection 9.2.1, discusses the R^2 performances of the ERV model after replacing empirical train samples with synthetic samples. The second part addresses the impact on the basic ERV model performance after augmenting empirical train samples with synthetic shell records in subsection 9.2.2.

9.2.1 Replacing empirical train samples

For all TVAE and CTGAN submodels, 50 synthetic samples[2] are used to train the ERV model, whereafter the R^2 performance is calculated on the empirical test set. Since the performance of the ERV model is dependent on the quality of a submodel that generated the synthetic samples, the ERV performance will be referred to as the submodel performance. The final submodel performance is the mean of the 50 calculated performances.

The TVAE and CTGAN submodel performances are compared to the basic ERV model performances. For each train sample size, the basic ERV model is trained 50 times on an empirical sample that is sampled from the total train set[3]. Hereafter, the final basic ERV model performance per train size is calculated by taking the mean of the 50 calculated performances. The black line in figure 9.6 represents the performances per train size of the basic ERV model. The basic ERV model performs best after being trained on the full empirical train set (69.852 records), having an $R^2 = 0.8471$. The worst performance of the basic ERV model was after being trained on the smallest sample size, resulting in a performance with $R^2 = 0.6429$. Clearly there is a positive relation between the performance of the basic ERV model and the train sample size.

[2]All 50 synthetic samples are independently generated
[3]The empirical train samples are sampled from the total train set in a stratified fashion on the feature "municipality"

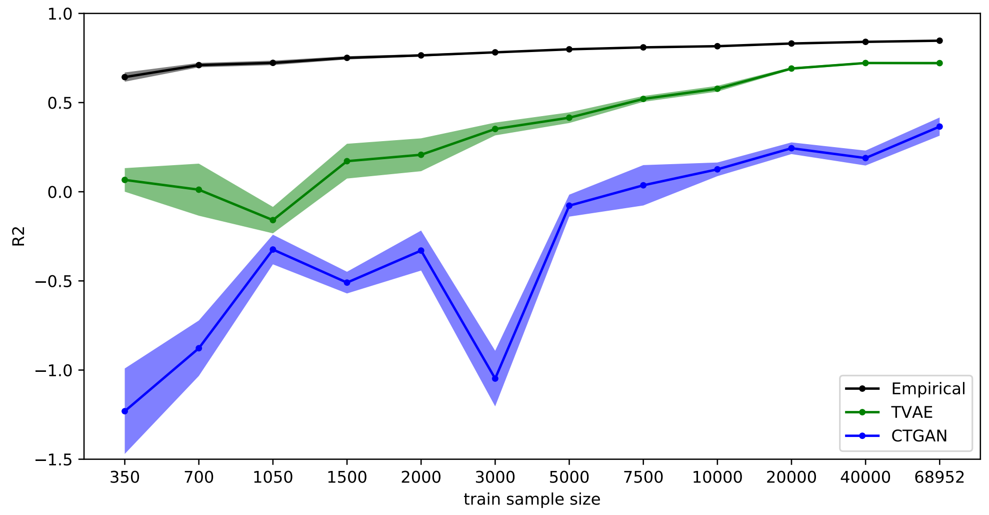


Figure 9.6: R^2 performances, with standard deviation, for the basic ERV model (black line), TVAE submodel performances (green line) and CTGAN submodel performances (blue line). Recall that submodel 1 is trained on sample size 350, submodel 2 on train sample size 700, up to submodel 12 on the full train sample of size 68.952.

TVAE

The TVAE submodel performances in figure 9.6 show a similar pattern as the TVAE similarity scores in figure 9.1. However, the performance for submodels trained on sample sizes 1.050 - 40.000 strictly increases and the performances for submodels trained on sample sizes 700 and 1.050 decrease and decrease more significantly, respectively, with respect to their predecessor submodels. Submodel 11, trained on sample size 40.000, is the best performing submodel with $R^2 = 0.7219$. This model deviates -14.12% from the best basic ERV model performance. The worst performing submodel is the submodel trained on sample size 1.050 with $R^2 = -0.1590$.

The green line clearly shows an increasing trend in R^2 when train samples are enlarged, indicating that the ERV model on average achieves a higher performance when being trained on synthetic data that is generated by a submodel for which a larger train sample is used.

CTGAN

Also the CTGAN submodels show a similar pattern for the R^2 performances in figure 9.6 as for the similarity scores in figure 9.1. The submodels trained on sample sizes 350 - 5.000 have a high variability in R^2 performances between consecutive submodels. Hereafter, the consecutive submodel performances increase, except for the submodel trained on sample size 40.000. The best performing CTGAN submodel is submodel 12, trained on sample size 68.952 , with $R^2 = 0.3654$. This submodel performance deviates 0.4816 from the best basic ERV model performance, which is a deviation of 43.43%. Submodels trained on sample sizes 350 - 5.000 all have a negative R^2 value, with submodel 1, trained on sample size 350, being the worst performing model with $R^2 = -1.2307$.

For the blue line, also an increasing trend is visible but starting at the submodel trained on sample size 3.000, indicating that the minimum required train sample size for the CTGAN model, to obtain an on average higher ERV model performance, is larger than for the TVAE model.

9.2.2 Augmenting empirical train samples

The results of this section can be divided into three parts. First, the impact on the basic ERV model performance on the shell test set is addressed after augmenting empirical train samples with 10.000 and 20.000 shell records[4], respectively. Second, the impact on the performances on furnished and upholstered test sets are discussed.

[4]The shell records are all sampled from TVAE submodels, since the TVAE model outperformed the CTGAN model in the statistical analysis and the first part of the machine learning efficacy analysis.

Finally, the impact on the performance on the total test set is discussed. The R^2 performances are calculated per train sample size and, just as in the first part of the machine learning efficacy analysis, are obtained by taking the mean of 50 calculated R^2 performances[5]. The figures in this section show the performance per sample size without standard deviation to keep the plots as clear as possible. However, each figure references in the description to the appendix section in which the figure is shown including standard deviation.

ERV performance on shell test set

The empirical train samples that are augmented are the samples with size 20.000, 40.000 and 68.952. The absolute performance shifts for all three sample sizes are listed in table 9.2.

augmentation size \train sample size	**20.000**	**40.000**	**68952**
+10.000 shell records	3.76	**4.91**	2.93
+20.000 shell records	4.42	**9.28**	5.87

Table 9.2: The change in R^2 performance of the basic ERV model on the shell test set, after augmenting empirical train samples with 10.000 and 20.000 shell records, respectively. The results are stated in percentages. The bold percentages indicate the largest performance improvement per augmentation size.

The performance of the basic ERV model increased for each train sample size after augmenting the train samples with 10.000 and 20.000 synthetic shell records, respectively. The largest increases was obtained by augmenting train samples of size 40.000. The increases in R^2 performance after augmenting the empirical train sample of size 40.000 with 10.000 and 20.000 shell records records were +4.91% and 9.28%, respectively. Figure 9.7 shows per train sample size the improvement of the basic ERV model after augmentation.

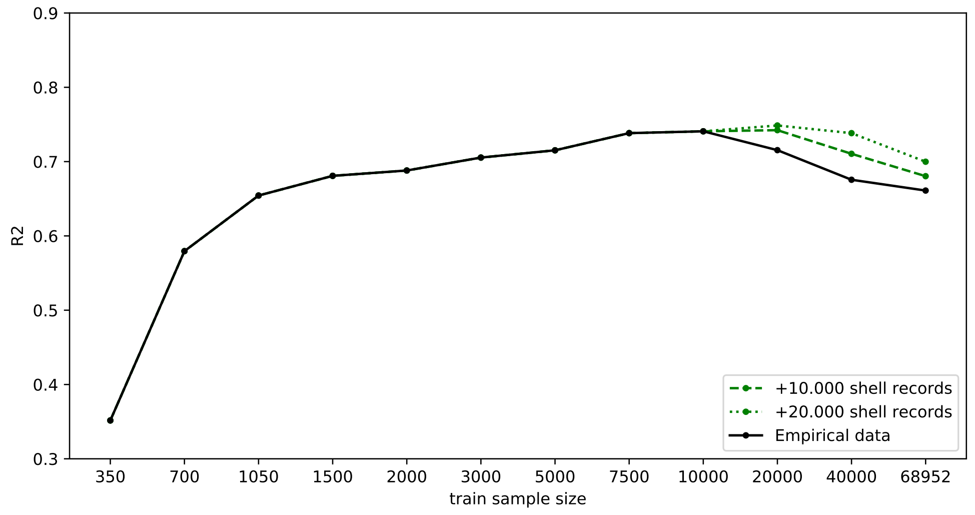


Figure 9.7: The R^2 performance per train sample size on the shell test set for the basic ERV model (black line), the basic ERV model after augmenting the empirical train samples with 10.000 shell records (green dashed line) and the basic ERV model after augmenting with 20.000 shell records (green dotted line). Remark that empirical train samples are augmented with shell records generated by a submodel that is trained on an equal size train sample. In subsection A.1.2, figure A.4 and A.5 show the mean performance, with standard deviation, of the basic ERV model after augmenting

ERV performance on furnished and upholstered test sets

In figure 9.8, the upper black line shows the basic ERV model performance on the upholstered test set. The green dashed and dotted lines are barely distinguishable from the black line. The same holds for the performance of the basic ERV model on the furnished test set, which is represented by the lower black line. Therefore it can

[5]The empirical train samples are independently sampled from the total train set in an stratified fashion. The shell samples of size 10.000 and 20.000 are sampled only once.

be concluded that augmenting the empirical train samples with shell records has no impact on the basic ERV model on furnished and upholstered test sets.

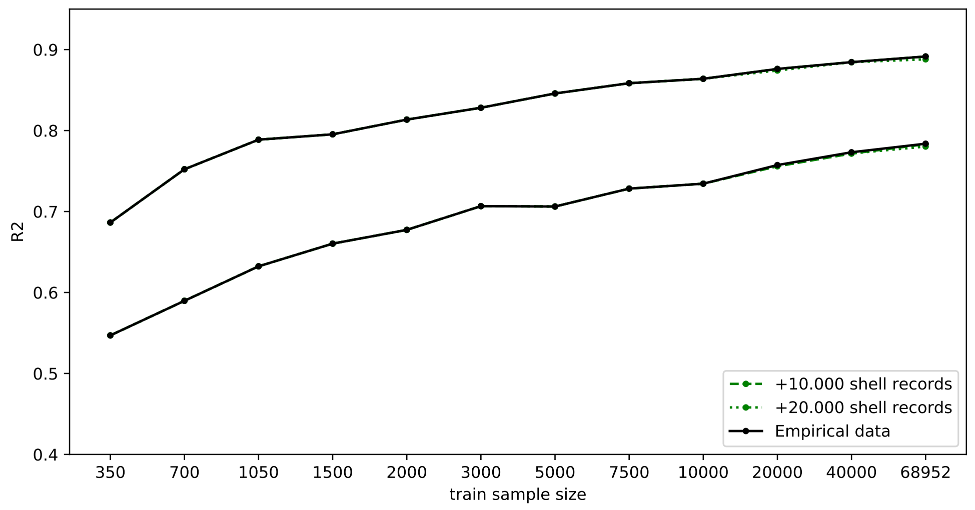


Figure 9.8: The R^2 performance per train sample size for the basic ERV model on the furnished test set (lower black line) and upholstered test set (upper black line), the basic ERV model after augmenting the train samples with 10.000 shell records (green dashed lines) and the basic ERV model after augmenting with 20.000 shell records (green dotted lines). Remark that empirical train samples are augmented with shell records generated by a submodel that is trained on an equal size train sample. In subsection A.1.2, the figures A.6 and A.7 show the performances on the total test set with standard deviation.

ERV performance on total test set

In figure 9.9, the green dashed and dotted lines are hardly distinguishable from the black line. This means that there is hardly any impact after augmenting the empirical train data with shell records on the basic ERV model performance on the total test set.

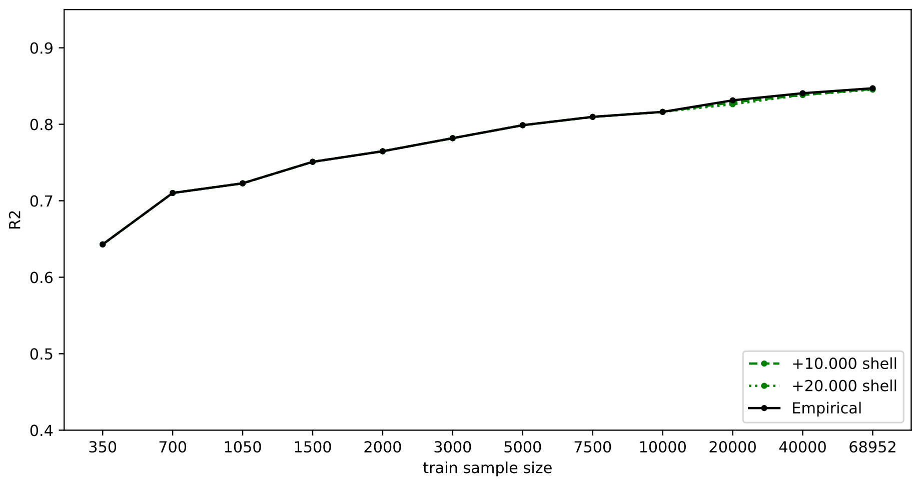


Figure 9.9: The R^2 performance per train sample size for the basic ERV model on the total train set (black line), the basic ERV model after augmenting the train samples with 10.000 shell records (green dashed line) and the basic ERV model after augmenting with 20.000 shell records (green dotted line). Remark that empirical train samples are augmented with shell records generated by a submodel that is trained on an equal size train sample. In subsection A.1.2, figures A.8 and A.9 show the performance on the total test set with standard deviation.

Conclusion

This research focussed on the question: *How well can deep generative models learn tabular real estate transaction data?*. The goal was to use two deep generative models, TVAE and CTGAN, to learn the underlying probability distribution function from which the empirical real estate transaction data originates and to sample synthetic real estate transaction data from that distribution. How well both models learned the underlying probability distribution is quantified by means of a statistical analysis and a machine learning efficacy analysis, both performed on by the models generated synthetic data. In this chapter, the research question will be answered by summarizing and comparing how well the models performed in both analyses.

In the statistical analysis it is analyzed per model how well statistical properties from the empirical real estate transaction data are reproduced in synthetic data, by means of the similarity score. The TVAE model outperformed the CTGAN model with a similarity score of 0.88 against a similarity score of 0.84, respectively. Thereby, compared to the CTGAN model, smaller deviations between similarity scores of consecutive submodels can be concluded for the TVAE model. On average, both models obtained a relatively higher similarity score when being trained on a larger train sample.

In the first part of the machine learning efficacy analysis it is investigated for both models how well synthetic real estate transaction data can be used to replace empirical train data for Clappform's ERV model. The ERV model shows, after using different size synthetic train samples from both models, a decrease in R^2 performance compared to the basic ERV model. The ERV model trained on synthetic samples originating from the TVAE model outperforms the ERV model trained on synthetic data generated by the CTGAN model with a performance of $R^2 = 0.7219$ against $R^2 = 0.3654$, respectively. Compared to the best performance of the basic ERV model with $R^2 = 0.8471$, this is a decrease of 14.78% and a decrease of 43.15%, respectively. The ERV model achieved higher R^2 performance when being trained on synthetic data generated by a deep generative model that is trained on larger train samples.

In the second part of the machine learning analysis it is shown how well synthetic data can be used to improve the performance of Clappform's ERV model by balancing the imbalanced feature "interior type". The best improvement of the basic ERV model on the shell test set was 9.28%, after augmenting the empirical train sample of size 40.000 with 20.000 synthetic shell records. Augmenting the empirical train samples had no influence on the performance of the ERV model on the furnished, upholstered and total test sets since the performance remained approximately the same as for the basic ERV model.

This research aimed to identify how well deep generative models can learn real estate transaction data. Based on a statistical analysis it can be concluded that the used deep generative models can be used to generate synthetic samples that have a maximum similarity score of 0.88. From the machine learning efficacy analysis it can be concluded that the deep generative models can be used to approximate the best basic ERV model performance of $R^2 = 0.8471$ with a deviation of -14.78%. In addition, augmenting the empirical train samples with synthetic shell records has little or no effect on the ERV model performance on the furnished, upholstered and total test set. However, augmenting the empirical train samples increases the ERV performance on the shell test set, with a maximal increase of 9.28%, indicating that the ERV model became actually better in predicting rental prices for residences that have an interior type shell.

Discussion & Recommendations

The results of this study are discussed in a more comprehensive way in subsection 11.1, together with several limitations that were encountered during this research in subsection 11.2. Recommendations are given throughout the discussion when applicable.

11.1 Interpretation of Results

11.1.1 Statistical Analysis

In the statistical analysis, the similarity scores show that the TVAE model outperforms the CTGAN model. This implies that the TVAE model is better in learning the empirical probability distribution of the RTD. It is important to emphasize that the statistical analysis only addresses, per sample size, how well statistical properties from the *empirical RTD* are reproduced in the synthetic samples. Since the total empirical train data is also a sample from the unknown probability distribution, only strong conclusions can be drawn about how well the models were able to reproduce properties from the *empirical probability distribution* of the RTD instead of the underlying distribution function.

Reproducing the empirical probability distribution of a data set, however, has several applications. Due to the regulation of the GDPR, privacy sensitive data cannot simply be shared with third parties. In case empirical data are privacy sensitive, a synthetic "replica" could be shared. Therefore, it is useful to create synthetic data that originates from the empirical probability distribution. More specifically, in this case it is not necessary for the synthetic data to follow the underlying distribution of the empirical data.

In comparison with the TVAE model, the similarity scores for the CTGAN model were more variable when trained on sample sizes smaller than 5.000 records. In general, deep generative models require large amounts of data for a sufficient train process. In addition, the train process of GANs is based on the adversarial minimax game between generator and discriminator. This implicit method of learning the probability distributions is unstable and might therefore lead more easily fail to converge. Due to the high variability in similarity scores, this model does not seem to converge when being trained on samples smaller or equal to 5.000. The TVAE model shows a less variable increasing trend for the similarity score from train sample size 1.050. This implies that this model is better in reproducing statistical properties from the RTD when trained on small sample sizes than the CTGAN model.

The small standard deviation in similarity score per train sample size could be explained by the fact that empirical samples are drawn from the total train set in a stratified fashion on the feature "municipality". This feature contains categories that represent a strong minority of the total train data. Therefore, when a relatively small empirical sample size is drawn multiple times in a stratified fashion, the sample will contain the same records from the minority categories of "municipality" with high probability. This reduces the variability in the train samples and leads directly to reduction of the variability of the similarity score. Thereby, the variability of empirical train samples decreases when larger train samples are drawn from the total train set. More specifically, when two or more samples are drawn that are larger than 50% of the total train set, the samples will contain equal records. This reduces the variability in the empirical samples and directly leads to reduction in variability in the similarity score. A suggestion for further research would be to sample randomly instead of in a stratified fashion. The samples on which the models are trained are already drawn in a stratified fashion. Randomly sampling increases variability in the samples. Therefore, it would be interesting to see how similar synthetic samples are to more variable empirical samples.

For both models a positive relation appears to exist between the similarity score and the sample size on which the models are trained. From this it can be concluded that training deep generative models with relatively large train samples results in better learning the empirical probability distribution of the RTD than when training on small samples.

11.1.2 Machine Learning Efficacy

In the first part of the machine learning efficacy analysis, the performance of the ERV model is analyzed per train sample size on the total test set after replacing empirical train samples with synthetic samples.

The ERV model trained on synthetic samples generated by the TVAE model achieves a higher R^2 performance for each train sample size than when being trained on synthetic data generated by the CTGAN submodel. This implies that the TVAE model was better in learning the underlying probability distribution of the real estate transaction data than the CTGAN model. This strong conclusion can be drawn since, in contrast to the statistical analysis, the performance of the ERV model is calculated on the unseen empirical test set. Therefore, conclusions can be made about how well the ERV model generalized. If a model generalizes well, it has high ability to adapt to new unseen (test) data, that is drawn from the same underlying distribution as the train data. The more similar the underlying probability distributions of (synthetic) train en (empirical) test data are, the better the ERV model is able to generalize.

However, the performance of the ERV model is, besides the quality of the synthetic train sample, also dependent on the choice of hyperparameters. For every performance calculation, the hyperparameters of the basic ERV model are used instead of fitting new hyperparameters to the synthetic train samples. Therefore, the obtained R^2 performances per train sample size might be suboptimal. Future research might extend the machine learning efficacy by cross validating the ERV model on synthetic data. In this way, a better quantification of the ERV performance can be quantified which leads into a more reliable indication of the quality of synthetic data.

The performance of the ERV model increases when being trained on larger synthetic train samples. This implies that deep generative models trained on large empirical train samples better learn the underlying probability distribution of the real estate transaction data than when being trained on small empirical train samples. The fact that both models perform poorly unless being trained on very large train samples might be caused by the size of the test set. This set contains 17.238 records. Small samples in general have less variability than large samples. If, in addition, the synthetic samples are of low quality, the variability in small samples might even be worse than in the small empirical train sample. Training the ERV model on small low quality synthetic samples might therefore lead to even less explained variability of the test set. A suggestion for future research would be redoing the experiment multiple times with several different sized test sets.

In the second part of the machine learning efficacy analysis, the impact of augmenting the empirical train samples with synthetic records on the ERV model performance on the shell, furnished, upholstered and total test set is analyzed. Only the ERV model performance on the shell test set increased. The largest increase was 9.28% after augmenting the empirical train sample of size 40.000 with 20.000 shell records. This implies that, after augmenting the empirical train samples with synthetic shell records, the ERV model became better at predicting the rental price for residences which have interior type shell. The second best improvement was an increase of 5.87% after augmenting the empirical train sample of size 68.952 with 20.000 shell records. This increase being the second best performance improvement on the shell test set could be explained by the fact that the TVAE model trained on empirical train sample with size 68.952 also performed second best in the statistical analysis.

The performance of the ERV model on the furnished, upholstered and total test set remained the same. Augmenting the empirical train samples with synthetic shell records has therefore no impact on the performance of the ERV model. However, the categorical feature ”interior type” is only one of the imbalanced features in the RTD. Balancing all imbalanced features with synthetic records would have given a better indication on how well the performance of the ERV model could have been improved with after augmenting the empirical train samples with synthetic records.

Future research could analyze how the performance of the ERV model on the total test set changes when balancing all imbalanced features in the real estate transaction data. In case it can be concluded that balancing multiple features has a positive impact on the ERV performance, an interesting research topic is to investigate how many synthetic records were necessary per unit of improvement of the ERV model and if there are imbalanced features that contribute more than other features to this improvement.

11.2 Limitations

11.2.1 Real Estate Transaction Data

Despite the RTD being a clean and already partly pre-processed data set, it has some limitations. First, the data are created by human entry. Therefore, the quality of the data may be debatable. Encountered flaws in the data were for example values for the feature ”floor level” that exceeded the value for feature ”number of floors building”. The same flaw was encountered for features ”total bedrooms” and ”total rooms”. Moreover, the data consisted of several categorical features that contained a significant percentage of missing values. This could also be a result of inaccurate human entry. In order to not increase the percentage of missing values, invalid records could not be removed and assumptions had to be made to modify the existing flaws. This might have had impact on the data quality.

To obtain better quality RTD, the data collection process could be improved. For example by giving explicit descriptions per feature for the data providing parties. Thereby, for features with relatively low percentage of missing values, imputation methods could be used to decrease the overall percentage of missing values.

11.2.2 Computational Power

Deep generative models are notorious for their long run time when taking large samples as input. The runtime for the CTGAN and TVAE models, for their given configuration and taking the full train sample (68.952 records) as input, was 23 and 16 hours, respectively, on a Macbook Pro. Thereby, training the CTGAN and TVAE model on all 12 different empirical train samples resulted in a total run time of 49 and 32 hours, respectively.

Because of these extremely long runtimes, both the TVAE and CTGAN model are trained on 12 empirical samples only once. This means that the results of both the statistical and the machine learning efficacy analysis might be dependent, to some extent, on these selected empirical train samples. In case sufficient GPU power is available, an interesting question for further research would be to investigate to what extent analysis results are dependent of the used empirical train samples.

In addition, for each train sample size, both models are trained with their corresponding configurations. Besides the quality of the input data, deep generative models are highly sensitive for the choice of hyperparameter values. However, the parameter space for deep generative models is unbounded since hyperparameters such as number of layers and number of nodes per layer can be made arbitrarily large. Exploring the hyperparameter space for an optimal configuration for both models and for each sample size was infeasible with the current available computational power and time. Therefore, the decision was made to train both models only on one configuration.

Future research could focus on optimization of hyperparameters for both models. Recent literature has shown that evolutionary algorithms can be used to improve hyperparameters values for deep neural networks in reasonable time. For example, the authors (Xiao et al., 2020) state that their method, a variable length genetic algorithm, can efficiently discover deep learning models with variable depths in reasonable time and makes hyperparameter optimization more feasible when there are limited computing resources available. The authors (Alarsan & Younes, 2021) show that genetic algorithms can be used to find good hyperparameters for a GAN architecture.

Appendix

A.1 Machine Learning Efficacy Analysis

A.1.1 Basic ERV Model Performance

This section shows the mean R^2 performance per train sample size, with standard deviation, of the basic ERV model on the shell, furnished and upholstered test sets in figures A.1, A.2 and A.3, respectively. Per train sample size, the mean performance is calculated by taking the average of 50 basic ERV performances. Each basic ERV model performance results after being trained on a newly sampled empirical train sample from the total train set.

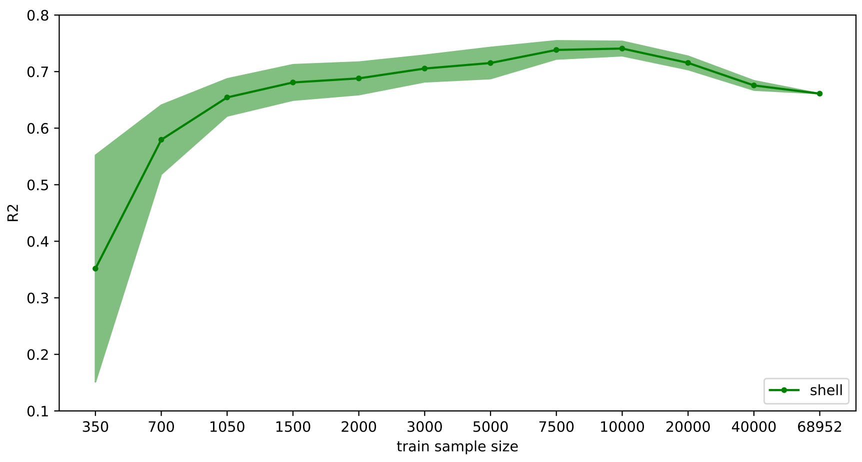


Figure A.1: Mean basic ERV model performance per train sample size, with standard deviation, on shell test set.

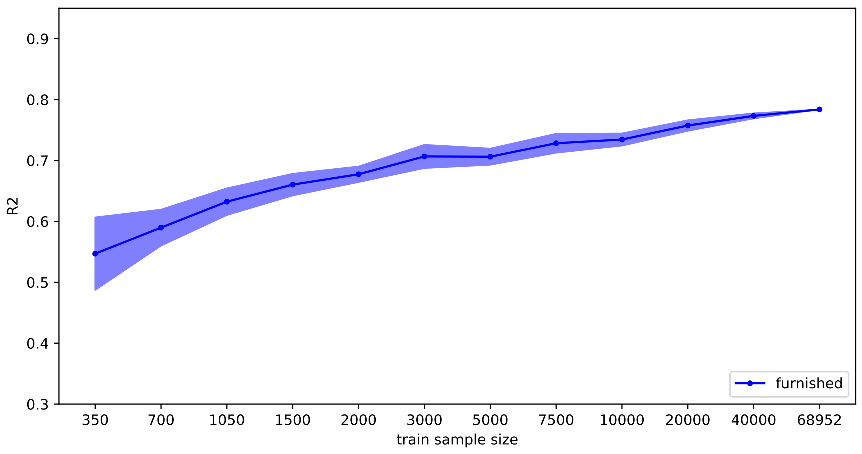


Figure A.2: Mean basic ERV model performance per train sample size, with standard deviation, on furnished test set.

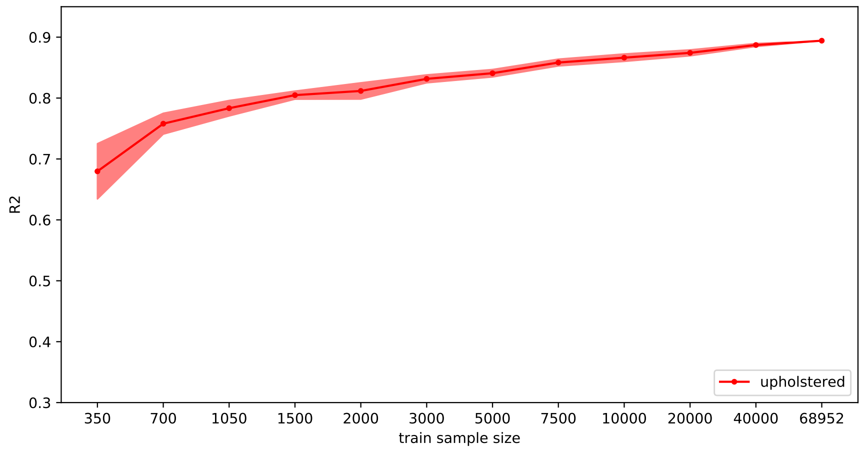


Figure A.3: Mean basic ERV model performance per train sample size, with standard deviation, on upholstered test set.

A.1.2 Basic ERV Model Performance after Augmenting

This section shows the mean R^2 performance per train sample size, with standard deviation, of the basic ERV model on the shell test set after augmenting the empirical train samples with synthetic shell records. Figure A.4 and A.5 show the mean performance after augmenting the train samples of size of 20.000, 40.000 and 68.952 with 10.000 and 20.000 shell records, respectively. In both figures, the mean performance of the basic ERV without augmenting is also shown with the black line. However, this performance with standard deviation can be found in figure A.1

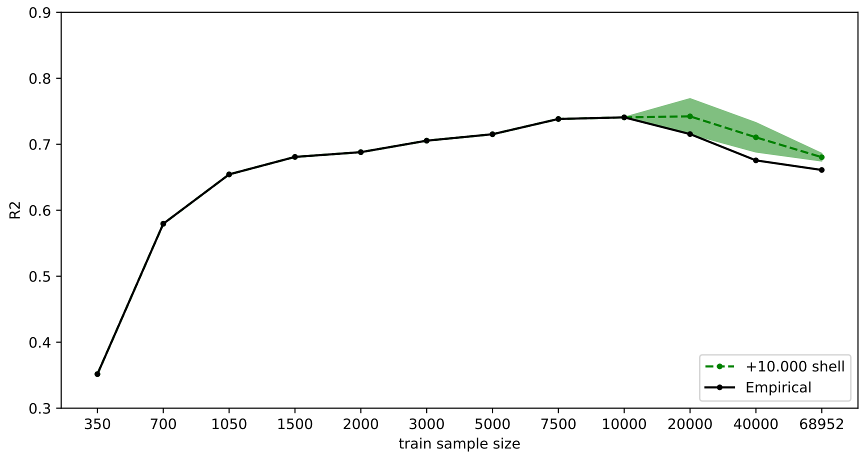


Figure A.4: Mean basic ERV model performance per train sample size, with standard deviation, on the shell test set after augmenting empirical train samples of size 20.000, 40.000 and 68.952 with 10.000 shell records.

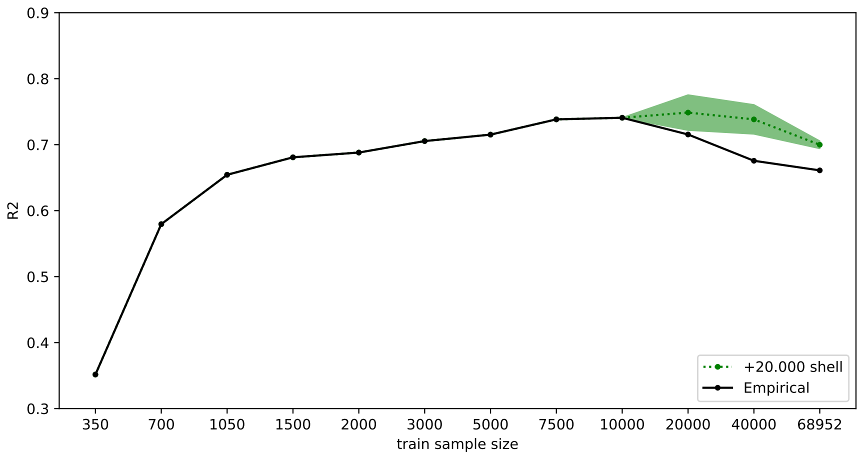


Figure A.5: Mean basic ERV model performance per train sample size, with standard deviation, on the shell test set after augmenting empirical train samples of size 20.000, 40.000 and 68.952 with 20.000 shell records.

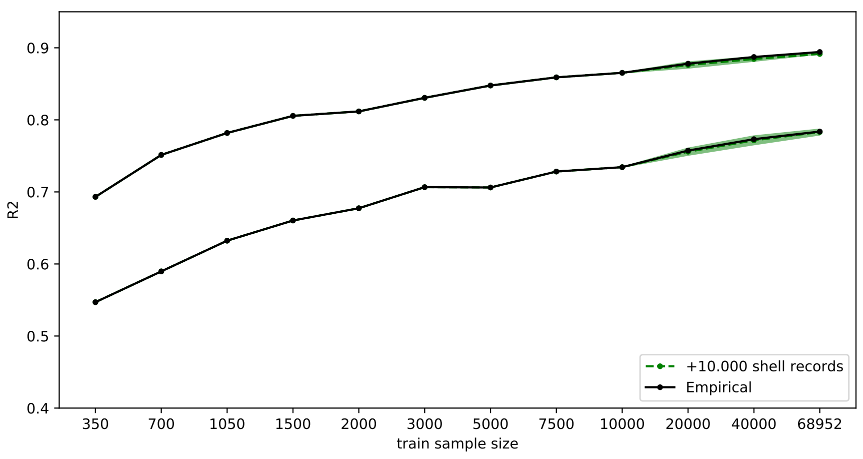


Figure A.6:

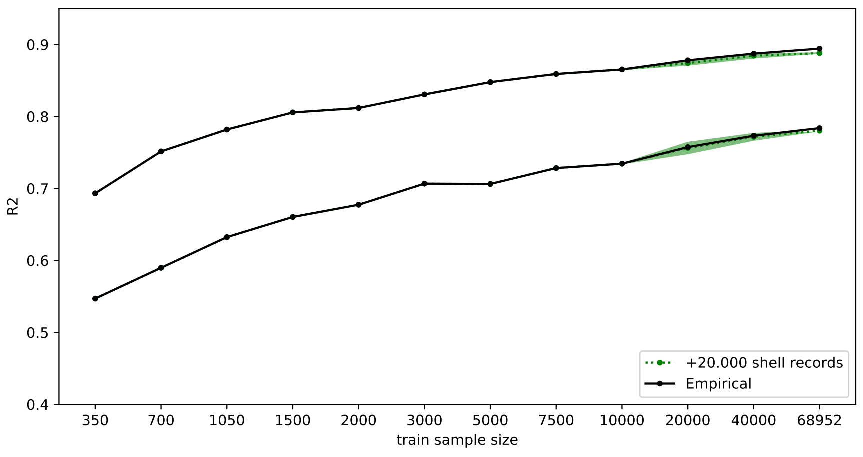


Figure A.7:

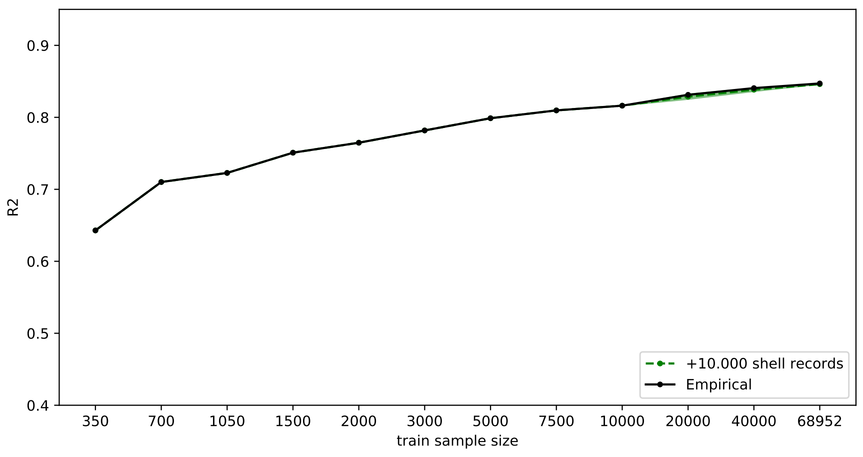


Figure A.8:

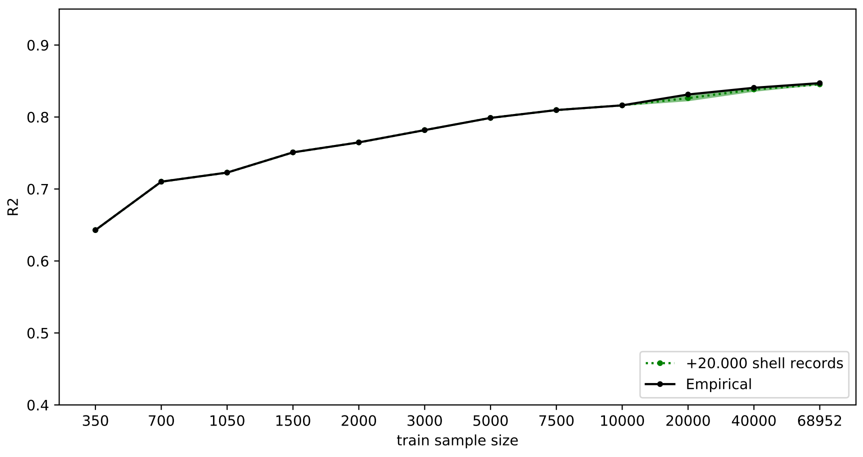


Figure A.9:

Bibliography

Abadi, M., Chu, A., Goodfellow, I., Mcmahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep Learning with Differential Privacy. https://doi.org/10.1145/2976749.2978318

Akoglu, H. (2018). User's guide to correlation coefficients. *Turkish journal of emergency medicine*, *18*(3), 91–93. https://doi.org/10.1016/j.tjem.2018.08.001

Alarsan, F. I., & Younes, M. (2021). Best selection of generative adversarial networks hyper-parameters using genetic algorithm. *SN Computer Science*, *2*(4), 1–14. https://doi.org/10.1007/s42979-021-00689-3

Amin-Naseri, M. R., & Soroush, A. R. (2008). Combined use of unsupervised and supervised learning for daily peak load forecasting. *Energy conversion and management*, *49*(6), 1302–1308. https://doi.org/10.1016/j.enconman.2008.01.016

Assefa, S. (2020). Generating synthetic data in finance: Opportunities, challenges and pitfalls. *Challenges and Pitfalls (June 23, 2020).*

Baak, M., Koopman, R., Snoek, H., & Klous, S. (2020). A new correlation coefficient between categorical, ordinal and interval variables with pearson characteristics. *Computational Statistics & Data Analysis*, *152*, 107043. https://doi.org/10.1016/j.csda.2020.107043

Bcom, R. K. P. (2012). Correlational analysis. *Australian Critical Care*, *25*, 195–199. https://doi.org/10.1016/j.aucc.2012.02.003

Berger, V. W., & Zhou, Y. (2014). Kolmogorov–smirnov test: Overview. *Wiley statsref: Statistics reference online.*

Bergh, D. (2015). Sample Size and Chi-Squared Test of Fit—A Comparison Between a Random Sample Approach and a Chi-Square Value Adjustment Method Using Swedish Adolescent Data. *Pacific rim objective measurement symposium (proms) 2014 conference proceedings* (pp. 197–211). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-47490-7{_}15

Bhatia, S. (2019). *Generative adversarial networks for improving imbalanced classification performance* (Doctoral dissertation). University of Dublin.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.*

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, *112*(518), 859–877.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., & Kasneci, G. (2021). Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889.*

Bourou, S., El Saer, A., Velivassaki, T.-H., Voulkidis, A., & Zahariadis, T. (2021). A review of tabular data synthesis using gans on an ids dataset. *Information*, *12*(9), 375. https://doi.org/10.3390/INFO12090375

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349.*

Brenninkmeijer, B., & Hille, Y. (2019). *On the Generation and Evaluation of Tabular Data using GANs* (Doctoral dissertation).

Chen, X., Pawlowski, N., Rajchl, M., Glocker, B., & Konukoglu, E. (2018). Deep Generative Models in the Real-World: An Open Challenge from Medical Imaging. http://arxiv.org/abs/1806.05452

Cho, J., Lee, K., Shin, E., Choy, G., & Do, S. (2015). How much data is needed to train a medical image deep learning system to achieve necessary high accuracy? *arXiv preprint arXiv:1511.06348.*

Choi, E., Biswal, S., Malin, B., Duke, J., Stewart, W. F., & Sun, J. (2017). Generating multi-label discrete patient records using generative adversarial networks. *Machine learning for healthcare conference*, 286–305.

Connelly, L. (2019). Chi-square test. *Medsurg Nursing*, *28*(2), 127–127.

Dahmen, J., & Cook, D. (2019). Synsys: A synthetic data generation system for healthcare applications. *Sensors*, *19*(5), 1181. https://doi.org/10.3390/s19051181

De Myttenaere, A., Golden, B., Le Grand, B., & Rossi, F. (2016). Mean absolute percentage error for regression models. *Neurocomputing*, *192*, 38–48. https://doi.org/10.1016/j.neucom.2015.12.114

Durgam, V. (2018). Social media and its role in marketing. *International Journal of Advanced Research in Management*, *9*(2). https://doi.org/10.4172/2151-6219.1000203

Eckelt, K., Adelberger, P., Zichner, T., Wernitznig, A., & Streit, M. (2019). Tourdino: A support view for confirming patterns in tabular data. *EuroVA@ EuroVis*, 7–11.

El Emam, K. (2020). Seven Ways to Evaluate the Utility of Synthetic Data. *IEEE Security and Privacy*, *18*(4), 56–59. https://doi.org/10.1109/MSEC.2020.2992821

Fisher, R. A. (1970). *Statistical methods for research workers*. Hafner Pub. Co.

Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., & Greenspan, H. (2018). Synthetic data augmentation using gan for improved liver lesion classification. *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, 289–293.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. http://arxiv.org/abs/1406.2661

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.

Harshvardhan, G., Gourisaria, M. K., Pandey, M., & Rautaray, S. S. (2020). A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, *38*, 100285. https://doi.org/10.1016/j.cosrev.2020.100285

Hawkins, D. M. (1980). *Identification of outliers* (Vol. 11). Springer. https://doi.org/10.1007/978-94-015-3994-4

Hittmeir, M., Ekelhart, A., & Mayer, R. (2019). On the utility of synthetic data: An empirical evaluation on machine learning tasks. *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 1–6. https://doi.org/10.1145/3339252.3339281

Hoogendoorn, M., & Funk, B. (2018). Machine learning for the quantified self. *On the art of learning from sensory data*.

Hu, X., Chu, L., Pei, J., Liu, W., & Bian, J. (2021). Model complexity of deep learning: A survey. *arXiv preprint arXiv:2103.05127*.

Hughes, G. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE transactions on information theory*, *14*(1), 55–63.

Ioffe, S., & Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift* (tech. rep.).

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

Kalisch, M., Michalak, M., Sikora, M., Wróbel, Ł., & Przystałka, P. (2015). Influence of outliers introduction on predictive models quality. *Beyond databases, architectures and structures. advanced technologies for data mining and knowledge discovery* (pp. 79–93). Springer. https://doi.org/10.1007/978-3-319-34099-9

Kaur, A., & Kumar, R. (2015). Comparative analysis of parametric and non-parametric tests. *Journal of computer and mathematical sciences*, *6*(6), 336–342.

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, *30*(1/2), 81–93.

Kim, K., & Hong, J.-s. (2017). A hybrid decision tree algorithm for mixed numeric and categorical data in regression analysis. *Pattern Recognition Letters*, *98*, 39–45. https://doi.org/10.1016/j.patrec.2017.08.011

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

Krishnan, P., & Jawahar, C. (2016). Generating synthetic data for text recognition. *arXiv preprint arXiv:1608.04224*.

Kukreja, V., Kumar, D., Kaur, A., et al. (2020). Gan-based synthetic data augmentation for increased cnn performance in vehicle number plate recognition. *2020 4th International Conference on Electronics, Com-*

munication and Aerospace Technology (ICECA), 1190–1195. https://doi.org/10.1109/ICECA49313.2020.9297625

Lantz, B. (2013). The large sample size fallacy. *Scandinavian journal of caring sciences*, *27*(2), 487–492. https://doi.org/10.1111/j.1471-6712.2012.01052.x

Lee, J., & Park, K. (2021). Gan-based imbalanced data intrusion detection system. *Personal and Ubiquitous Computing*, *25*(1), 121–128.

Lehmkuhl, L. D. (1996). Nonparametric statistics: Methods for analyzing data not meeting assumptions required for the application of parametric tests. *JPO: Journal of Prosthetics and Orthotics*, *8*(3), 105–113.

Li, F., Zhang, L., Chen, B., Gao, D., Cheng, Y., Zhang, X., Yang, Y., Gao, K., Huang, Z., & Peng, J. (2018). A light gradient boosting machine for remainning useful life estimation of aircraft engines. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 3562–3567.

Ma, C., Tschiatschek, S., Hernández-Lobato, J. M., Turner, R., & Zhang, C. (2020). Vaem: A deep generative model for heterogeneous mixed type data. *arXiv preprint arXiv:2006.11941*.

Maaløe, L., Fraccaro, M., Liévin, V., & Winther, O. (2019). Biva: A very deep hierarchy of latent variables for generative modeling. *arXiv preprint arXiv:1902.02102*.

Malandrakis, N., Shen, M., Goyal, A., Gao, S., Sethi, A., & Metallinou, A. (2019). Controlled text generation for data augmentation in intelligent artificial agents. *arXiv preprint arXiv:1910.03487*.

Massey Jr, F. J. (1951). The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, *46*(253), 68–78.

McHugh, M. L. (2013). The chi-square test of independence. *Biochemia medica*, *23*(2), 143–149.

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., et al. (2019). Evolving deep neural networks. *Artificial intelligence in the age of neural networks and brain computing* (pp. 293–312). Elsevier.

Mohamed, S., & Lakshminarayanan, B. (2016). Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*.

Nasteski, V. (2017). An overview of the supervised machine learning methods. *Horizons. b*, *4*, 51–62.

Nazabal, A., Olmos, P. M., Ghahramani, Z., & Valera, I. (2020). Handling incomplete heterogeneous data using vaes. *Pattern Recognition*, *107*, 107501. https://doi.org/10.1016/j.patcog.2020.107501

Nguyen, G. H., Bouzerdoum, A., & Phung, S. L. (2008). A supervised learning approach for imbalanced data sets. *2008 19th international conference on pattern recognition*, 1–4.

Nikolenko, S. I. (2021). Synthetic data for deep learning.

Omar, N., Sengur, A., & Al-Ali, S. G. S. (2020). Cascaded deep learning-based efficient approach for license plate detection and recognition. *Expert Systems with applications*, *149*, 113280. https://doi.org/10.1016/j.eswa.2020.113280

Park, N., Mohammadi, M., Gorde, K., Jajodia, S., Park, H., & Kim, Y. (2018). Data synthesis based on generative adversarial networks. *arXiv preprint arXiv:1806.03384*. https://doi.org/10.14778/3231751.3231757

Pearson, K. (1896). Mathematical contributions to the theory of evolution. III. Regression, Heredity, and Panmixia. *Philosophical Transactions of the Royal Society of London. Series A*, *187*, 253–318. https://doi.org/https://doi.org/10.1098/rsta.1896.0007

Pearson, R. K. (2002). Outliers in process modeling and identification. *IEEE Transactions on control systems technology*, *10*(1), 55–63.

Puri, G. D., & Haritha, D. (2016). Survey big data analytics, applications and privacy concerns. *Indian Journal of Science and Technology*, *9*(17), 1–8. https://doi.org/10.17485/ijst/2016/v9i17/93028

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rankin, D., Black, M., Bond, R., Wallace, J., Mulvenna, M., & Epelde, G. (2020). Reliability of supervised machine learning using synthetic data in health care: Model to preserve privacy for data sharing. *JMIR Medical Informatics*, *8*(7), e18910. https://doi.org/10.2196/18910

Ruthotto, L., & Haber, E. (2021). An introduction to deep generative modeling. *GAMM-Mitteilungen*, e202100008.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, *27*(3), 379–423.

Shwartz-Ziv, R., & Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, *81*, 84–90.

Srivastava, N., Hinton, G., Krizhevsky, A., & Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting* (tech. rep.).

Wagner, P. (2020). Privacy enhancing technologies and synthetic data. *Available at SSRN 3762686*. https://doi.org/10.2139/SSRN.3762686

Walters-Williams, J., & Li, Y. (2009). Estimation of mutual information: A survey. *International Conference on Rough Sets and Knowledge Technology*, 389–396.

Wan, Z., Zhang, Y., & He, H. (2017). Variational autoencoder based synthetic data generation for imbalanced learning. *2017 IEEE symposium series on computational intelligence (SSCI)*, 1–7.

Wen, B., Colon, L. O., Subbalakshmi, K., & Chandramouli, R. (2021). Causal-tgan: Generating tabular data using causal generative adversarial networks. *arXiv preprint arXiv:2104.10680*.

Wu, Q., Gao, R., & Zha, H. (2019). Stein bridging: Enabling mutual reinforcement between explicit and implicit generative models. *arXiv preprint arXiv:1909.13035*.

Xiao, X., Yan, M., Basodi, S., Ji, C., & Pan, Y. (2020). Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm. *arXiv preprint arXiv:2006.12703*.

Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). Modeling tabular data using conditional gan. *arXiv preprint arXiv:1907.00503*.

Xu, L., & Veeramachaneni, K. (2018). Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264*.

Zafarani, R., Abbasi, M. A., & Liu, H. (2014). *Social media mining: An introduction*. Cambridge University Press.