Lund University

Self-supervised learning on tabular data

An investigation into different implementations of VIME

Bachelor thesis

Tova Jahnke
Supervised by Mattias Ohlsson

June 5, 2024

Contents

Abstract

With the objective to classify a tabular data set of breast cancer patients with a high accuracy the self-supervised model VIME [1] is studied. The influence of several hyperparameters during pre-training is investigated and AUC of the downstream task is regarded as the measurement of performance. A larger unlabeled synthetic data set is generated using the Synthetic Data Vault (SDV) [2]. Different sizes is then pre-trained on and the result evaluated in the downstream task. Using synthetic data gives result of similar standard to the original set. Moreover an alternative mask generator implementing the correlations between features using two different methods is proposed. Both methods produce effective results compared to the original stochastic version and have arguably great potential for further research.

Populärvetenskaplig beskrivning

Det har varit svårt att missa uppsvinget av den allmänna tillgången till AI de senaste åren. Juridiken, arbetsmarknaden och industrin behöver alla anpassa sig fort för att hänga med. Med chattbotar och generativ AI som kan skapa en bild föreställande precis vad som helst på bara några minuter är det svårt att inte bli imponerad. I media har det uttryckts fascination men även en slags rädsla för hur framtiden ska se ut. Vad som inte är lika välkänt är den artificiella intelligens vi redan har inkorporerat i våra liv. Äger du en smartphone med ansiktsigenkänning eller en självkörande bil har du redan stått öga mot öga med AI. Musik- och översättningsappar är ytterligare två exempel.

En del av artificiell intelligens kallas maskininlärning vilket är ett paraplybegrepp för olika metoder för att träna modeller och algoritmer till att bli ännu bättre med minimal direkt hjälp från människor. Dessa modeller kan bli otroligt duktiga på att hantera stora mängder data och hitta mönster och korrelationer. Detta kan i sin tur leda till effektiva förutsägelser om framtida data.

Antalet utbildade och kunniga dataspecialister ökar snabbt men det gör även efterfrågan på det de kan producera. Därmed är det viktigt att kunna bygga och träna modeller med så lite mänsklig interaktion som möjligt. Större nätvärk och datamängder ger generellt fördelaktigare resultat men detta till en kostnad då de kräver mer tid och arbete. Därför sysslar dataingenjörer och forskare med att på olika sätt optimera modeller. Ett exempel är ett relativt nytt koncept som kallas självövervakat lärande. Detta är ett koncept som kräver minimal mänsklig interaktion samtidigt som det kan ge bättre resultat än helt oövervakade motsvarigheter. Dessa modeller har huvudsakligen byggts med avseende på bilder och texter. På senare år har dock även tabulär data börjat användas. En stor fördel med denna typ av lärande är att det inte krävs lika mycket annoterad data (där svaret man ska hitta är känt, så att man kan träna direkt på det) som andra typer. Stora mängder annoterad data kan vara tidskrävande och därmed kostandsdrivande att ta fram. Därför har förslag för att effektivisera självövervakat lärande på tabulär data varit fokus i många artiklar på senare tid och även i den här.

1 Introduction

Machine learning is a branch of artificial intelligence (AI) which has been greatly advanced in recognition and applications in the past years. Due to the great technological advances of the world a considerable amount of information is stored digitally. Machine learning models of all types have been developed and is it an increasingly growing field. Existing models implement all types of data such as images, tabular data and texts. Their mutual goal is to find patterns or correlations in the input data and make relevant predictions by optimizing weights and hyperparameters during training [3].

As a general rule larger models as well as data sets tend to generate more accurate results [4]. There are however disadvantages to these attributes. Computational requirements and time are two factors which can be negatively affected. Therefore other ways of improving the results with a smaller cost are often explored. In this paper various propositions are made and analyzed with the aim to advance the progress made by an existing self-supervised learning model named VIME and postulated by Yoon [1]. Using data representing medical features of breast cancer patients the influence of hyperparameters in VIME is analyzed. Additionally the consequences of using different sizes of a larger synthetic data set for pre-training is studied. A method implementing the correlations between the medical features is explored with the hope to advance the general model.

1.1 Self-supervised learning

Supervised learning is a method of training algorithms using a labeled dataset which is purposely divided for training, validation and testing. Characteristic supervised tasks include classification and regression. The labels can be seen as a type of categorization. In real-world examples the existence of labels for larger data sets is sparse and the process of labelling is an expensive one in terms of time and resources. Due to the manual process potential generalization errors can not be ignored. Therefore supervised learning is not always the preferred method.

In contrast, unsupervised learning trains using no labels. Self-supervised learning (SSL) can be described as a subsection of unsupervised learning since it usually is not given external labels for training. Self-supervised learning is a process where a model learns features from unlabeled data as a type of pre-training which then improves the training of labeled data. Largely unlabeled data sets can then be utilized and in that sense it can be argued that this process is superior to supervised learning. In particular, the general objective is to find relevant representations from data without labels. This is a relatively new field and most models are developed for data in image or text form. The results have been highly successful and preferable compared to supervised models. Several studies have been made on self-supervised learning with images. One task that has been the focus of several papers is the jigsaw puzzle. In simple terms an image is divided into a number of tiles which are then rearranged. A network then tries to solve the puzzle by finding the original image as a pretext task. This process generates information about the image which then can be assistive in a down-stream task. This particular approach was implemented by Noroozi and Favaro in 2017 [5]. They concluded that due to the pretext task the network becomes competent in identifying the tiles as parts of the whole image as well as their placements. The results can be argued to be a great advancement in self-supervised learning and an indicator that the field should be further explored.

The use of self-supervised models on tabular data is however less explored yet has been effective in published studies. Hajiramezanali et al.in 2022 [6] proposes STAb which implements self-supervised representation learning on tabular data with stochastic regularization. Another effective implementation is VIME [1] which will be introduced in the following section.

1.2 VIME

VIME was introduced in 2020 by Yoon, J. [1] as a proposition for advancing the field of self-supervised learning, on tabular data in particular. The model can be described as a systematic framework which takes advantage of large unlabeled data sets. By pre-training and producing a smaller representation of the data the accuracy of a downstream task, where a smaller amount of labeled data is used, can be optimized.

There are different ways of pre-training self-supervised networks. One method which is relevant for this study can be described by an encoder function denoted by $e : \mathcal{X} \to \mathcal{Z}$ which uses an unlabeled data sample $\mathbf{x}$ as input and creates a latent representation $\mathbf{z}$ as a function of the input. It contains meaningful information of the data accompanied by so called pseudo-labels $y_s \in \mathcal{Y}_s$. This representation is then used in a so called downstream task.

A sample of an unlabeled dataset $\mathbf{x} \in \mathcal{D}_u$ together with a mask vector $\mathbf{m} = [m_1, ..., m_j]^\intercal$ is used as input into a pretext generator g_m. The mask is generated with the same dimensions as $\mathbf{x}$ and its elements m_i take on randomly distributed binary values from a Bernoulli distribution with a probability p_m which is used as a hyper parameter. In the pretext generator the mask vector now acts as a guide to which values should be masked by being layered on top of the data sample, $g_m : \mathcal{X} \times \{0,1\}^d \to \mathcal{X}$. The values chosen by $\mathbf{m}$ are interchanged with other values $\bar{\mathbf{x}}$ in their respective columns which will generate a corrupted sample $\tilde{\mathbf{x}}$ as described below.

$$\tilde{\mathbf{x}} = g_m(\mathbf{x}, \mathbf{m}) = \mathbf{m} \odot \bar{\mathbf{x}} + (1 - \mathbf{m}) \odot \mathbf{x} \tag{1}$$

The output of the pretext generator is used as an argument to the encoder e which transforms it into a representation of the latent space denoted by $\mathbf{z}$. The randomness of the generator determines the difficulty of reconstructing $\mathbf{x}$ from the latent representation and can be modified by increasing or decreasing the parameter p_m. The reconstruction is rendered in two discrete steps. Yoon proposes two MLPs that perform mask vector estimation and feature vector estimation, respectively, as pretext tasks.

The mask vector estimator s_m uses the latent representation to find a vector which should describe which features have been corrupted $s_m : \mathcal{Z} \to [0,1]^d$. This is represented in a vector $\hat{\mathbf{m}} = (s_m \circ e)(\tilde{\mathbf{x}})$. In a similar way the feature vector estimator s_r instead tries to predict the actual values of the sample $\mathbf{x}$, its output the vector $\hat{\mathbf{x}}$.

A loss function l_m is defined as the sum of the binary cross-entropy losses for the dimensions of the mask vector $\mathbf{m}$.

$$l_m(\mathbf{m}, \hat{\mathbf{m}}) = -\frac{1}{d}[\sum_{j=1}^{d} m_j \log\left[(s_m \circ e)_j(\tilde{\mathbf{x}})\right] + (1 - m_j) \log\left[1 - (s_m \circ e)_j(\tilde{\mathbf{x}})\right]] \tag{2}$$

Moreover a reconstruction loss l_r is introduced.

$$l_r(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{d}[\sum_{j=1}^{d}(x_j - (s_r \circ e)_j(\tilde{\mathbf{x}}))^2] \tag{3}$$

When training the entire model including the encoder e as well as the two estimators s_m and s_r simultaneously the objective becomes to optimize the losses. Finally a combined optimization task is constructed, where the encoder and the estimators are trained.

$$\min_{e, s_m, s_r} \mathbb{E}_{\mathbf{x} \sim p_X, \mathbf{m} \sim p_{\mathbf{m}}, \tilde{\mathbf{x}} \sim g_m(\mathbf{x}, \mathbf{m})}[l_m(\mathbf{m}, \hat{\mathbf{m}} + \alpha \cdot l_r(\mathbf{x}, \hat{\mathbf{x}})]$$

A parameter to adjust the weight of the reconstruction loss is denoted as α.

The encoder and its latent representation can then be used in a downstream task. The objective is generally to make $\mathbf{z}$ contain as much relevant information as possible. Subsequently the latent representation $\mathbf{z}$ is arguably the most important part of the pre-training and the main goal should be to optimize it by maximizing quality while keeping the size small.

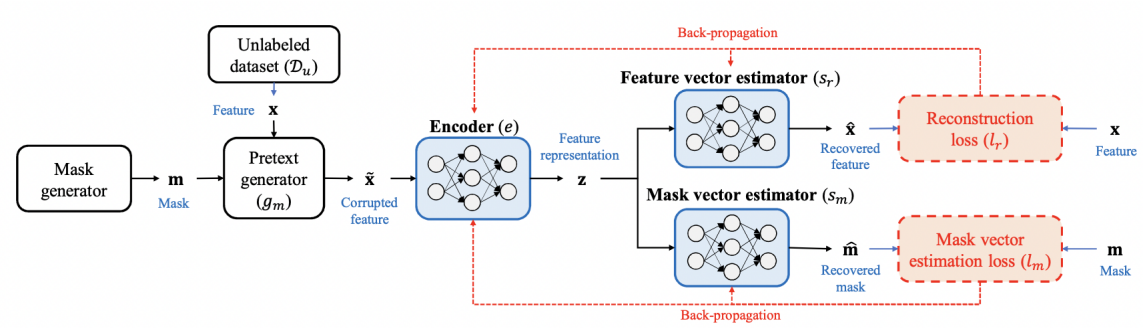


Figure 1: Diagram taken from the article describing the VIME model [1]. The data set as well as the generated mask are combined in the pretext generator where the masked values are corrupted and this is sent into the encoder. Furthermore the two MLPs built to reconstruct the corruption is given the output of the encoder and returns the feature and mask with some loss.

2 Method

The basis for this project is VIME and the general objective is to find potential improvements. For transparency the main code can be found on GitHub[1]. As an initial task different hyperparameters are tested when pre-training on the breast cancer data (see section 4.1). Primarily the influence of the structure and size of the two estimators as well as the size of the latent representation is evaluated. The result is determined through the value of the AUC in the downstream task. For accurate results the average value over five runs through the entire pipeline is calculated along with the standard deviation.

The synthetic data set is evaluated in a similar way where the hyperparameters are modified while producing results. Moreover, different sizes of the data set are considered separately and their results compared. This part of the investigation is done to be able to understand the quality of the synthetic data as well as if larger data sets make a difference.

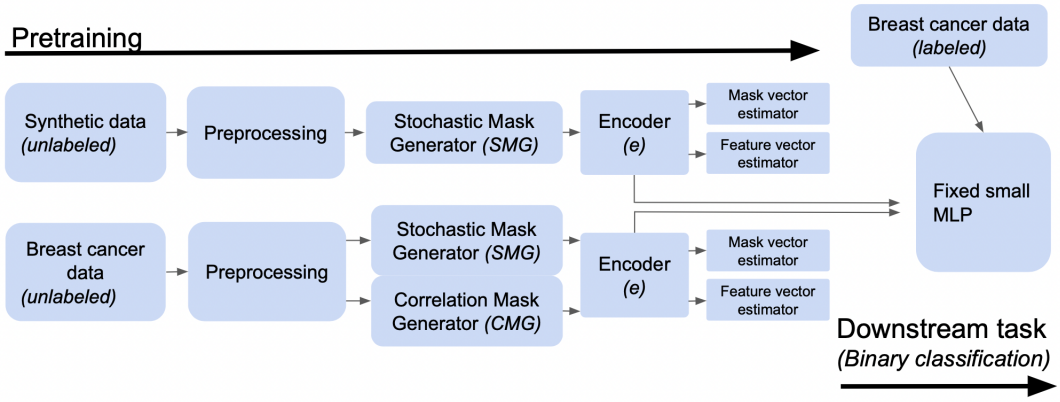


Figure 2: In this figure the uppermost part represents the task of analyzing the synthetic data and the lower one instead corresponds to the original breast cancer data. This lower part incorporates two different paths for the mask generator. However, every path arrive at the same MLP for the downstream task.

When having an understanding of the impact of model structure the mask generator is remodeled. The correlation between the features of the breast cancer data set is taken into account and the choice of which values to corrupt is implemented. This is done with the intention to improve the amount of relevant information in

[1] https://github.com/TovaJahnke/Self-supCMG

the latent representation that is sent to the network performing the downstream task. The complete pipeline is depicted in Figure 2.

2.1 Breast cancer data

The original data set studied in this report is referred to as breast cancer data and contains information about women with a breast cancer diagnosis received at Skåne University Hospital located in Lund, Sweden between January 2009 and December 2012. Information about preceding results related to the diagnoses was collected from the Swedish National Quality Registry for Breast cancer as well as public mammography screening program records. The general features were instead obtained from medical records. The data was collected and studied by Dihge et al. in 2019 [7]. There are 800 rows which represent the amount of patients as well as 27 columns with medical features describing the patients. Each patient has been assigned a binary label "N0" which confirms or denies the find of metastases in the respective patients lymph nodes. This is conveyed through a binary value.

In total the columns chart 9 continuous and 12 categorical features. The continuous ones include tumor size, degree of severeness and general medical features such as age, height, weight and BMI. There are 4 columns describing the status of progesterone and estrogen in the patient's body, two of these are binary and the others are continuous. The last continuous feature Ki67 is a measure of how quickly the cancer cells are dividing [8].

Furthermore, there are five columns representing the position ("C500", "C502", "C503", "C504" or "C505") of the main tumor. They can collectively be regarded as one feature and therefore the mask generator has been designed such that if one of the position columns are corrupted so are the other ones. The same is true for the three binary columns representing the particular tumor diagnosis since it can be classified as either 1, 2 or greater than 2.

The binary feature "Mammography screening" corresponds to if the patient has been diagnosed using this method or not. Other binary features relay each patient's menopause status, if more than one tumor has been found as well as if the cancer has been found in both breasts or not, respectively. If unilateral, there is a feature named "Side" which conveys in which breast the tumor has been found.

It is useful to train models on this specific data set since it can lead to classification with a high accuracy which in turn could help surgeons decide if the risk of a more complicated surgery is worth taking or not. Generally surgeons check for metastases during surgery for removing the main tumor. This process does however involve risks in damaging important nerves in the patient's arm. The main goal is to give them more information to make educated decisions without having to perform complex surgeries.

2.2 Synthetic data

An additional data set is generated for the purpose of improving the results. This is a synthetic data set which is produced as a synthetic alternative based on the original breast cancer data. As a result the eligibility of this method can be evaluated.

To generate the synthetic data set the Python library Synthetic Data Vault (SDV) [2] is utilized and the code can be found in Appendix A (5). It was created at the MIT Data to AI lab in the year of 2016 and has since progressed to DataCebo which is the current developer. The SDV library contains several different synthesizers which have been developed for different purposes although with the mutual objective of creating synthetic data through recognizing patterns in the original data. For this particular case the Tabular Variational Autoencoder (TVAE) [9] was applicable. The "N0" label is removed from the breast cancer set. In the first part, the encoder compresses the data into a representation in the latent space with dimensions that are defined in the code. This is achieved by creating one mean vector and another standard deviation vector. These are in turn utilized to generate a random latent vector $\mathbf{z}_s$ [10]. A sample of these vectors is then given to the decoder which will generate an output. For each of these samples the TVAE is trained using

back-propagation. In Table 1 the values of the hyperparameters are shown. The loss which is a combination between reconstructing and regularization loss between the inputs and outputs is compared and missing data is handled in the synthesizer. The number of epochs and the batch size are optimized during training. Lastly the output is a generated matrix of data with similar properties to the original which in this report is referred to as the synthetic data set.

The generated data set consists of 100 000 rows of artificial patients and different subsets of this is used when training to be able to analyze the impact of larger sets as well as the quality of the synthetic data.

Hyperparameters	Values
Number of epochs	5000
Batch size	75
L2-scale	$1 \cdot 10^{-3}$
Loss factor	2
Size of encoder hidden layer	(70, 50)
Size of decoder hidden layer	(50, 70)

Table 1: List of hyperparameters and their respective values of the TVAE when generating the synthetic data.

2.3 Preprocessing data

The breast cancer data $\mathcal{D}^{bc}$ and the synthetic data $\mathcal{D}^{synth}$ are both normalized using z-score normalization as a part of the data preprocessing part seen in Figure 2. This is a normalization method which uses the mean μ_x and standard deviation σ_x in the following way, where $\mathbf{x} \in \mathcal{D}$ is a column in the raw data and $\mathbf{x}'$ its normalized counterpart.

$$\mathbf{x}' = \frac{\mathbf{x} - \mu_x}{\sigma_x} \tag{4}$$

2.4 Masking

For the synthetic data pre-training as well as the upper pipeline of the breast cancer data in Figure 2 the labels are removed from the original data set and the data points are put into two different mask generators. The Stochastic Mask Generator (SMG) creates a binary matrix $\mathbf{m}$ the size of the unlabeled data set $\mathcal{D}_u$ it is given. The placements of ones and zeros are random and their ratio p_m is a hyperparameter of the encoder as proposed in VIME [1]. The synthetic data set is only applied using this method. The masked values are then corrupted through being substituted with another value of the respective column.

With the purpose of analyzing the effect of implementing the feature correlations a Correlation Mask Generator (CMG) is constructed. It is initialized using the same stochastic method as for the Stochastic Mask Generator (SMG). Once again p_m i.e the ratio of masked values is treated as a hyperparameter. To determine which features should be considered correlated a correlation coefficient is calculated between every feature, pairwise. This is calculated using Pearson correlation with the exception of comparing two binary features. In this case Matthews correlation is applied. Pairs that produce a correlation coefficient higher than or equal to the absolute value of 0.5 are treated as correlated. Since the mask vector is layered on top of the data sample each of its elements is mapped one-to-one to each element in the data sample. Therefore it is now possible to augment the mask vector to take the correlations into account. This is done using two different approaches;

(1) *The values of the generated mask vector corresponding to the elements of a correlated pair are put equal to each other.*

(2) *The values of the generated mask vector corresponding to the elements of a correlated pair are ensured to*

not be equal to each other.

Since the ratio of 1s and 0s is modified the representative parameter p_m is now recalculated and the new one is referred to as p_m^{new}.

Formally, two features A and B considered correlated are put into a correlation matrix $\mathcal{M}5$. For each row i in $\mathcal{M}$ one of the two elements are chosen using a random choice method. The chosen element will be considered the pair's primary $P_i = A_i \vee B_i \, \forall i$. The secondary value S_i is then left. For approach (1) the mask vector value of the primary element is what the secondary value's mask vector value is set to as well, $m_{S,i} = m_{P,i}$. In approach (2) the secondary's mask vector value is put to one minus the primary's, $m_{S,i} = |1 - m_{P,i}|$.

$$\mathcal{M} = \begin{bmatrix} A_1 & B_1 \\ A_2 & B_2 \\ \vdots & \vdots \\ A_i & B_i \end{bmatrix} \tag{5}$$

The encoder stack of the pre-training model converts the now corrupted data to the latent representation $\mathbf{z}$. Its architecture is kept constant with one hidden layer consisting of 27 nodes.

2.5 Downstream task

The downstream task is the same for all pre-training models. The objective is to classify the breast cancer data into two classes. These classes corresponds to the binary "N0"-label. The encoder created during pre-training is now used for the downstream task. Its parameters are frozen to make sure they are not a part of training once again. For all models the labeled breast cancer data set $\mathcal{D}_l$ is used for testing with the same split of the data used in the pre-training.

The MLP used for the binary classification is linear with no hidden layers (i.e. a perceptron). The is on behalf of understanding the influences of the different implementations of the pretraining without any increase in performance due to the model of the downstream task. Using a more advanced MLP may produce a more accurate result that is not an effect of the representation learning.

In a similar way to the pre-training the test loss is plotted and observed. The measurement used to evaluate performance is AUC.

2.6 Implementation details

The Multilayer Perceptrons (MLPs) performing the mask vector and feature estimations use the latent representation as input and have one hidden layer each. The amount of nodes h_s in these layers are uniformly modified throughout the experiment. During each respective run 30% of the data is used for validation and the remaining 70% for training. The specific split is kept constant using *seed* = 10. The validation loss is noted and visualized as a function of steps using Tensorboard [11]. The graph is studied to ensure convergence and to rule out overfitting. Additional parameters which are kept constant are shown in Table 2.

Model	Batch size	Learning rate	α	Epochs
Pre-training	400	0.0001	3.0	3000
Downstream	200	0.0001	-	5500

Table 2: Hyperparameters held constant during training.

3 Result and Discussion

3.1 Breast cancer data

In Table 3 the results of the downstream task when using a pre-trained encoder of different sizes are shown. They are also illustrated in Figure 3 where the standard deviations can be observed in the error bars. The variable h_s represents the amount of hidden nodes in the estimators. The size of the latent representation is denoted by z_s. The values are averages taken of five consecutive runs through the whole chain. It is quite clear that using a very small latent representation gives unclear and undesirable results. A z_s at 27 or above gives better results, in particular in combination with more hidden nodes.

h_z	$z_s = 2$	$z_s = 10$	$z_s = 16$	$z_s = 27$	$z_s = 35$
2	0.51	0.60	0.64	0.66	0.69
8	0.56	0.66	0.70	0.71	0.72
27	0.58	0.73	0.74	0.75	0.78
35	0.57	0.73	0.74	0.76	0.77

Table 3: Average AUC values of five consecutive executions of the downstream task using an encoder pre-trained on the breast cancer data.

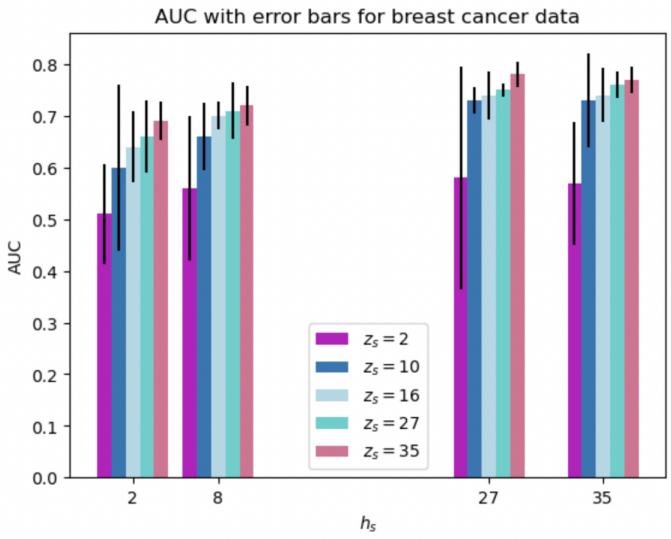


Figure 3: Average AUC for different sizes of hidden nodes and latent representation for the breast cancer data with the standard deviations shown as error bars.

.

It can be argued that the AUC seem to increase with increasing latent representation size z_s. Values above 35 of z_s were not tested since the primary objective of this task was to find optimal hyperparameters for implementing the synthetic data as well as the Correlation Mask Generator (CMG). Due to this it is not of interest to find the absolute best model in terms of which one gives the highest AUC. Conversely the smallest models in terms of z_s and h_s may not produce unbiased results in the other tasks due to their limited capability to separate data and store relevant information, respectively. With this in mind the choice of structures to use when analyzing the impact of the hyperparameter p_m as well as the different implementations of the mask generators is made. Putting both sizes z_s and h_s to 27 is the first selection and $z_s, h_s = (10, 8)$ is the second.

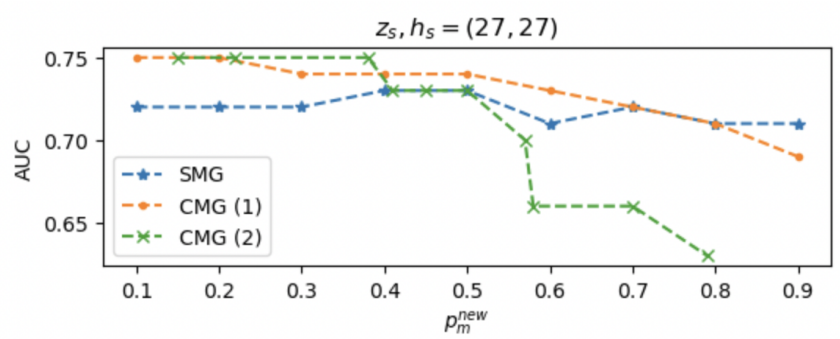
When analyzing the influence of the different mask generators a range of the hyperparameter p_m was tested.

Since it is defined as the ratio of masked values an updated value p_m^{new} was calculated when using the Correlation Mask Generator (CMG). These values are tabulated in Table 4. For method (1) defined in Section 4.4 it was approximately unaffected. However, the same can not be said for method (2). For the lower half of the range the new ratio is greater than the original value. When reaching 0.50 the ratios are essentially the same and for the upper half the trend reverses. Furthermore, when p_m is put to zero p_m^{new} becomes 0.15.

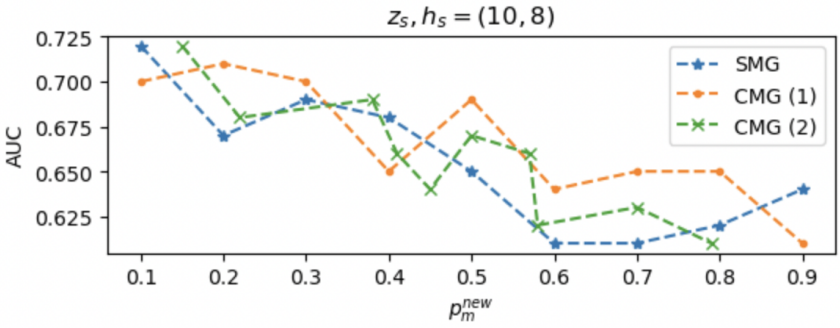
p_m	p_m^{new} for CMG (1)	p_m^{new} for CMG (2)
0.00	≈ 0.00	≈ 0.15
0.10	≈ 0.10	≈ 0.22
0.20	≈ 0.20	≈ 0.38
0.30	≈ 0.30	≈ 0.41
0.40	≈ 0.40	≈ 0.45
0.50	≈ 0.50	≈ 0.50
0.60	≈ 0.60	≈ 0.57
0.70	≈ 0.70	≈ 0.58
0.80	≈ 0.80	≈ 0.70
0.90	≈ 0.90	≈ 0.79

Table 4: Tabulated values of the hyperparameter p_m and the approximative values of its updated counterpart p_m^{new}, for the two methods using the CMG.

The AUC values of the downstream task obtained when pre-training using the different mask generators are shown in Figure 4. Analyzing these results the first conclusion to be made is that the defined correlations of features in the breast cancer data indeed affect the results. This is a fair assertion when observing the variations in results when incorporating the Correlation Mask Generator (CMG). When using the original SMG it can be expected for the AUC to decrease with increasing p_m since it essentially increases the difficulty and randomness for the estimators. Observing the result for the smaller sizes in part (b) of Figure 4 such a pattern can be confirmed. Though it could be stated that both or either one of the hyperparameters h_s and z_s are too great for p_m to make a significant difference in the larger model since the behavior of its result is almost uniform.



(a)



(b)

Figure 4: An illustration of the relationship between the updated ratio p_m^{new} and AUC, for two different combinations of z_s and h_s. The tabulated version along with standard deviations is found in Appendix B (5).

It is clear that the larger sizes of both z_s and h_s are superior to the smaller ones in regards to having predominantly better and more stable results. In part (b) of Figure 4 the values of the AUC notably fluctuate, in particular for the two methods using the CMG. This is to be expected when taking into account that an increased number of nodes is beneficial in general. It could also be argued that when applying the CMG a larger model is increasingly important.

For lower values of the ratio the two methods of the CMG are superior to their stochastic counterpart. To hypothesise the cause of the elevated result of method (2) for smaller p_m^{new} it could be argued that if only one feature value of a correlated pair is masked the other could assist in the mask and feature estimation since their correlation still can be easily found by observing other values of the same feature columns.

Interestingly, the result of the CMG method (2) decreases significantly when reaching a p_m^{new} of around 0.5. An explanation for this could be that when more values are initially masked and one of the elements of a correlated pair is made sure to be changed the estimators may not have enough uncorrupted values to find these correlations. This would account for the decrease as well as the dissimilar behavior of the other implementations in the same illustration.

The method (1) result does not exhibit this fast decreasing behavior. Instead it follows a gradual downward slope. It could be argued that making sure all correlated pairs are either both masked or both not masked should make it harder for the estimators to find the mask vector and corrupted features. As a result it could be that the encoder finds a representation of the data that otherwise would be overlooked in lieu of the defined correlated features.

Based on these results it could be said that the implementation of correlated features was successful for lower p_m in regards to the comparison of the SMG in part (a) of Figure 4.

3.2 Synthetic data

For simple comparison with the breast cancer data set the synthetic data set was first applied with 800 data points. The downstream result after pre-training on this synthetic data subset can be seen in Fig 5. Overall the AUC increases moderately for bigger h_s and z_s. Another observation is the significant fluctuations when having less hidden nodes, as illustrated by the error bars. This is also true for smaller z as the deviations decrease for larger latent representations.

When observing the results for larger sizes of synthetic data it can be observed that the AUC increases when going up to 2000 and 5000 data points but at 10 000 there is no clear increase. Moreover, the standard deviations increase with larger data sets for h_s equal to 2 as well as 8 as seen in Figures 6, 7 and 8.

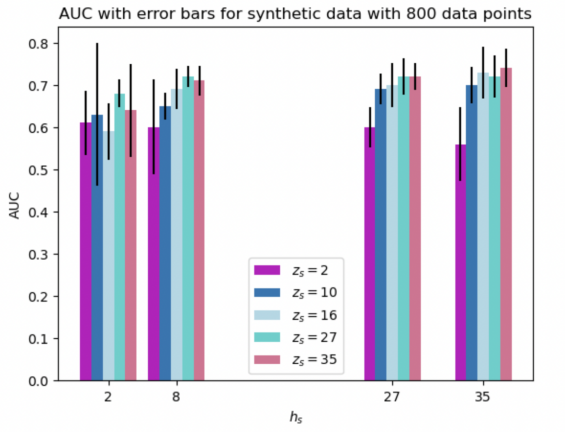


Figure 5: AUC for different sizes of hidden nodes and latent representation for 800 data points of synthetic data.

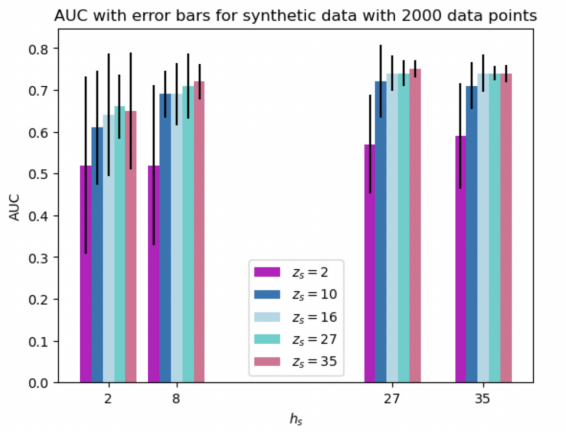


Figure 6: AUC for different sizes of hidden nodes and latent representation for 2000 data points of synthetic data.

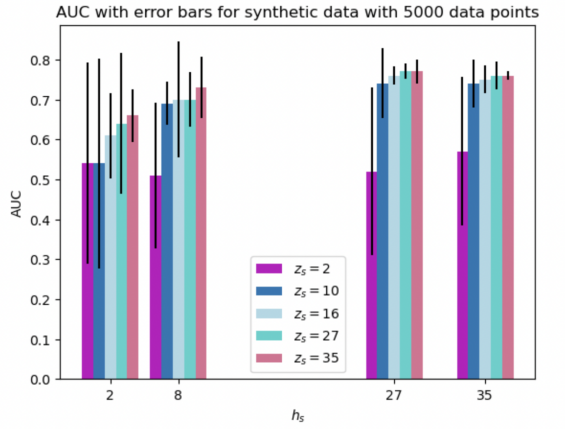


Figure 7: AUC for different sizes of hidden nodes and latent representation for 5000 data points of synthetic data.

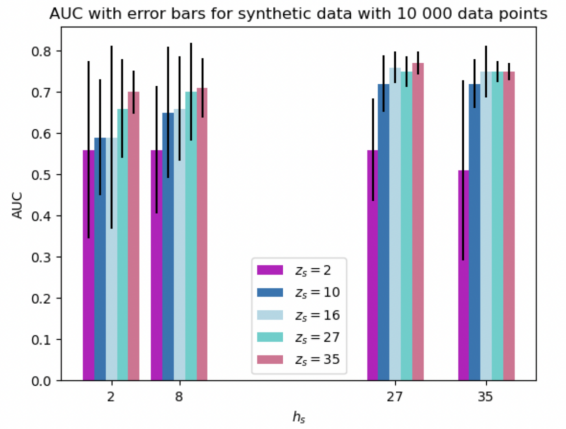


Figure 8: AUC for different sizes of hidden nodes and latent representation for 10 000 data points of synthetic data.

For 10 000 data points the highest AUC is 0.77 which was achieved with $z_s = 35$ and $h_s = 27$. This makes sense when comparing to the value for the same model with the breast cancer data in Table 3. The same AUC was achieved with 5000 data points for both $z_s, h_s = (27, 27)$ and $z_s, h_s = (35, 27)$ which was the highest for this subset as well. When going down to 2000 data points the highest result was 0.75 and this for $z_s, h_s = (35, 27)$. The subset that stands out in this regard is the smallest one where the highest AUC was 0.74 for $z_s, h_s = (35, 35)$. This means that 800 data points of the synthetic data is not enough to give results of the same standard as the ones for the breast cancer data. It could be argued that the quality of

the synthetic data is not as good as the original it is based on. However it should be mentioned that it is not a significant difference, based on this argument.

Lu et al. argues in their review of synthetic data generation in machine learning [12] that synthetic healthcare data can be beneficial when considering patient confidentiality. If the generated data can imitate the distribution of the original data the privacy of each patient can be preserved. It should be mentioned that this is not only advantageous for ethical reasons but for juridical as well. In Sweden there is the Patient Data Act [13] which limits the access to such information and similar laws are currently in place in other parts of the world. Using this as an argument the results of this investigation can be seen as successful. Even though it has not been observed that the synthetic data generates significantly preferable results it can not be said to be inferior to the breast cancer data when implementing enough data points.

4 Conclusion and outlook

The proposed self-supervised model VIME [1] was studied and potential improvements were proposed in this thesis. A breast cancer data set consisting of a number of patients with their respective medical features was used as well as a generated synthetic counterpart. The correlations between these features were utilized in the mask generator determining which values to corrupt.

In conclusion the VIME model works well on the breast cancer data implemented in this study. This is also true for the synthetic data generated and tested. Using enough data points gives a result of similar accuracy as for the original. This is a highly valuable result and can be particularly beneficial when dealing with medical data.

Moreover, interesting and favorable results were achieved when implementing correlations of features in the pre-training. A significant increase in performance was seen, in particular when initially masking less then half of the data points. It may also be stated that further investigation into different larger models would be beneficial when implementing correlations. Another approach could be to in more depth analyze the correlations between the features. Even though Pearson correlation seems to work relatively well it is linear which means it only picks up on one type of relationship. One example could be using the maximal information coefficient (MIC) [14] which has the ability to find a wider range of correlations.

5 Acknowledgements

A thank you to Max Svensson for granting permission to his code on GitHub[2] which was helpful for getting started with VIME. Another big thanks to Mattias Ohlsson for supervising this project!

[2]https://github.com/msvenssons/Mix-Encoder

List of abbreviations

AUC - Area Under Curve

BMI - Body Mass Index

CMG - Correlation Mask Generator

ER - Estrogen Receptor

MIC - Maximal Information Coefficient

MLP - Multilayer Perceptron

PR - Progesterone Receptor

SMG - Stochastic Mask Generator

TVAE - Tabular Variational Autoencoder

Appendix A

```
    # 1. Loading the data
    import pandas as  pd
from sdv.datasets.local import load_csvs

# assume that my_folder contains a CSV file named 'guests.csv'
datasets = load_csvs(
    folder_name='./',
    read_csv_parameters={
        'skipinitialspace': True,
        'encoding': 'utf-8',
        'sep': '\t'
    })

# the data is available under the file name
data = datasets['dataNO_NA-all-toSytheticML']
data.head()


from sdv.metadata import SingleTableMetadata

metadata = SingleTableMetadata()

metadata.detect_from_dataframe(data)

#metadata.visualize()
metadata

# 2. Creating a synthesizer

An SDV **synthesizer** is an object that you can use to create synthetic data. It learns patterns from t
from sdv.lite import SingleTablePreset
from sdv.single_table import TVAESynthesizer

synthesizer = TVAESynthesizer(
    metadata, # required
    enforce_min_max_values=True,
    enforce_rounding=True,
    epochs=5000,
    batch_size=75,
    compress_dims = (70,50),
    decompress_dims = (50,70),
    embedding_dim = 20,
    l2scale = 1E-3,
    loss_factor = 2
)

#synthesizer = SingleTablePreset(
#    metadata,
#    name='FAST_ML'
#)
synthesizer.fit(
    data=data
```

```
)

# Save for later use
#synthesizer.save('my_synthesizer.pkl')

losses = synthesizer.get_loss_values()

# Plot VAE loss
plot = losses['Loss'].plot(title='Training loss')

# Uncomment if we should use the save one
#synthesizer = SingleTablePreset.load('my_synthesizer.pkl')

NGen = 10000
synthetic_data = synthesizer.sample(
    num_rows= NGen
)

# save the data as a CSV
#synthetic_data.to_csv('dataN0-Synth-NaN.csv', sep = '\t', na_rep = 'NaN', index=True)

synthetic_data


# 4. Evaluating real vs. synthetic data

SDV has built-in functions for evaluating the synthetic data and getting more insight.

from sdv.evaluation.single_table import run_diagnostic

diagnostic = run_diagnostic(
    real_data=data,
    synthetic_data=synthetic_data,
    metadata=metadata
)

from sdv.evaluation.single_table import evaluate_quality

quality_report = evaluate_quality(
    data,
    synthetic_data,
    metadata
)

quality_report.get_details('Column Shapes')
from sdv.evaluation.single_table import get_column_plot

fig = get_column_plot(
    real_data=data,
    synthetic_data=synthetic_data,
    column_name='Tumörlokalisation',
#    column_name='Tumörstorlek',
#    column_name='ER',
    metadata=metadata
```

```
)

fig.show()
from sdv.evaluation.single_table import get_column_pair_plot

fig = get_column_pair_plot(
    real_data=data,
    synthetic_data=synthetic_data,
    column_names=['Tumörstorlek', 'BMI'],
    metadata=metadata
)

fig.show()

# 5. Saving and Loading
We can save the synthesizer to share with others and sample more synthetic data in the future.

synthesizer.save('my_synthesizer.pkl')

synthesizer = SingleTablePreset.load('my_synthesizer.pkl')
```

Appendix B

$z_s, h_s = (27, 27)$	
p_m	AUC
0.1	0.72 ± 0.01
0.2	0.72 ± 0.02
0.3	0.72 ± 0.02
0.4	0.73 ± 0.02
0.5	0.73 ± 0.03
0.6	0.71 ± 0.03
0.7	0.72 ± 0.03
0.8	0.71 ± 0.05
0.9	0.71 ± 0.05

Table 5: Average AUC values when using the Stochastic Mask Generator (SMG) for $z_s, h_s = (27, 27)$ and varying p_m.

$z_s, h_s = (10, 8)$	
p_m	AUC
0.1	0.72 ± 0.02
0.2	0.67 ± 0.03
0.3	0.69 ± 0.03
0.4	0.68 ± 0.05
0.5	0.65 ± 0.04
0.6	0.61 ± 0.05
0.7	0.61 ± 0.05
0.8	0.62 ± 0.06
0.9	0.64 ± 0.05

Table 6: Average AUC values when using the Stochastic Mask Generator (SMG) for $z_s, h_s = (10, 8)$ and varying p_m.

$z_s, h_s = (27, 27)$		
p_m^{org}	p_m^{new}	AUC
0.1	≈ 0.1	0.75 ± 0.02
0.2	≈ 0.2	0.75 ± 0.02
0.3	≈ 0.3	0.74 ± 0.02
0.4	≈ 0.4	0.74 ± 0.03
0.5	≈ 0.5	0.74 ± 0.02
0.6	≈ 0.6	0.73 ± 0.04
0.7	≈ 0.7	0.72 ± 0.05
0.8	≈ 0.8	0.71 ± 0.05
0.9	≈ 0.9	0.69 ± 0.06

Table 7: Average AUC values when using method (1) with the Correlation Mask Generator (CMG) for $z_s, h_s = (27, 27)$ and varying p_m.

$z_s, h_s = (10, 8)$		
p_m^{org}	p_m^{new}	AUC
0.1	≈ 0.1	0.70 ± 0.06
0.2	≈ 0.2	0.71 ± 0.05
0.3	≈ 0.3	0.70 ± 0.06
0.4	≈ 0.4	0.65 ± 0.06
0.5	≈ 0.5	0.69 ± 0.07
0.6	≈ 0.6	0.64 ± 0.07
0.7	≈ 0.7	0.65 ± 0.08
0.8	≈ 0.8	0.65 ± 0.07
0.9	≈ 0.9	0.61 ± 0.07

Table 8: Average AUC values when using method (1) with the Correlation Mask Generator (CMG) for $z_s, h_s = (10, 8)$ and varying p_m.

$z_s, h_s = (27, 27)$		
p_m	p_m^{new}	AUC
0.00	≈ 0.15	0.75 ± 0.02
0.10	≈ 0.22	0.75 ± 0.02
0.20	≈ 0.38	0.75 ± 0.03
0.30	≈ 0.41	0.73 ± 0.03
0.40	≈ 0.45	0.73 ± 0.03
0.50	≈ 0.50	0.73 ± 0.05
0.60	≈ 0.57	0.70 ± 0.06
0.70	≈ 0.58	0.66 ± 0.06
0.80	≈ 0.70	0.66 ± 0.05
0.90	≈ 0.794	0.63 ± 0.06

Table 9: Average AUC values when using method (2) with the Correlation Mask Generator (CMG) for $z_s, h_s = (27, 27)$ and varying p_m.

$z_s, h_s = (10, 8)$		
p_m^{org}	p_m^{new}	AUC
0.00	≈ 0.15	0.72 ± 0.04
0.10	≈ 0.22	0.68 ± 0.08
0.20	≈ 0.38	0.69 ± 0.06
0.30	≈ 0.41	0.66 ± 0.04
0.40	≈ 0.45	0.64 ± 0.05
0.50	≈ 0.50	0.67 ± 0.04
0.60	≈ 0.57	0.66 ± 0.06
0.70	≈ 0.58	0.62 ± 0.05
0.80	≈ 0.70	0.63 ± 0.05
0.90	≈ 0.79	0.61 ± 0.05

Table 10: Average AUC values when using method (2) with the Correlation Mask Generator (CMG) for $z_s, h_s = (10, 8)$ and varying p_m.

References

[1] J. Yoon, Y. Zhang, J. Jordon, and M. van der Schaar, "Vime: Extending the success of self- and semi-supervised learning to tabular domain," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 11 033–11 043. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/7d97667a3e056acab9aaf653807b4a03-Paper.pdf

[2] "The synthetic data vault. put synthetic data to work!" https://sdv.dev/, accessed: 10 May 2024.

[3] S. Brown, "Machine learning, explained," https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained, Apr. 2021, accessed: 2024-05-14.

[4] J. B. Simon, D. Karkada, N. Ghosh, and M. Belkin, "More is better in modern machine learning: when infinite overparameterization is optimal and overfitting is obligatory," 2024.

[5] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," 2017.

[6] E. Hajiramezanali, N. L. Diamant, G. Scalia, and M. W. Shen, "STab: Self-supervised learning for tabular data," in *NeurIPS 2022 First Table Representation Workshop*, 2022. [Online]. Available: https://openreview.net/forum?id=EfR55bFcrcI

[7] L. Dihge, M. Ohlsson, P. Edén, P.-O. Bendahl, and L. Rydén, "Artificial neural network models to predict nodal status in clinically node-negative breast cancer," *BMC Cancer*, vol. 19, p. 610, 2019. [Online]. Available: https://doi.org/10.1186/s12885-019-5827-6

[8] U. S. government, "NCI Dictionaries — cancer.gov," https://www.cancer.gov/publications/dictionaries, [Accessed 16-05-2024].

[9] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," in *Advances in Neural Information Processing Systems*, 2019.

[10] P. Edén and M. Ohlsson, *Introduction to Artificial Neural Networks and Deep Learning.* Lund, Sweden: Computational Science for Health and Environment (COSHE) - Lund University, 2023.

[11] "Tensorboard," https://www.tensorflow.org/tensorboard, accessed: 06 May 2024.

[12] Y. Lu, M. Shen, H. Wang, X. Wang, C. van Rechem, T. Fu, and W. Wei, "Machine learning for synthetic data generation: A review," 2024.

[13] Stockholm County Council, "The patient data act," https://www.slso.regionstockholm.se/en/Safeandsecurecare/Lawsandregulations/#:~:text=The%20Patient%20Data%20Act.,-The%20rules%20for&text=Among%20other%20things%2C%20this%20law,written%20by%20another%20healthcare%20organisation, 2023, accessed: 2024-05-15.

[14] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, "Detecting novel associations in large data sets," *Science*, vol. 334, no. 6062, pp. 1518–1524, 2011. [Online]. Available: https://doi.org/10.1126/science.1205438