Exploring The Use of Autoencoders With Multimorbidity Data

Jac Powell
210839106
William March
MSc Data Science and AI

Abstract

***Introduction* - This paper explores the use of autoencoders and variational autoencoders with multimorbidity data for the purposes of dimensionality reduction and synthetic data generation. While electronic health records have resulted new large, rich datasets, the 'curse of dimensionality' describes how difficult datasets with large dimensions can be to analyse.**

***Dataset* - The data used for this study are electronic health records of 40469 patients with a range of 2 or more of 204 long-term conditions. The dataset was encoded as binary variables, 0 if they had not been diagnosed with the condition and 1 if they had.**

***Standard Autoencoder Analysis* – Experiments were conducted on 3 aspects of the standard autoencoder. *(i)* Hidden layers *(ii) Batch Size (iii)* Dimensions of the latent layer. They were evaluated by analyzing the reconstruction loss and the correlation of the latent variables. The study found the best architecture and carried it forward. It saw that as batch size increased, the reconstruction reduced, however the correlation between the latent variables increased. As the number of latent variables increased the reconstruction loss reduced.**

***Variational Autoencoder Analysis* – The study found that the VAE did not perform as well as the standard AE with regards to reconstruction loss, however, the latent variables were independent. A beta weighting term for KL divergence was introduced. As the beta reduced so did the reconstruction loss, however, the correlation of the latent variables increased.**

***Clustering* – The data was reduced to 25 latent variables. K-means clustering was used with K=3 and it was found that all three clusters showed different conditions in their top 10 most prevalent conditions.**

***Synthetic Data Generator* – Synthetic data was generated using the means and standard deviations of the latent variables created in clustering. Initial comparison of the descriptive statistics between the synthetic data and reconstructed data was similar. The synthetic data were assigned based on the clustering algorithm fitted to real data. One of the clusters showed similar disease prevalence to real data.**

***Conclusion* – This study found that AE and VAE show promise as a form of dimensionality reduction for multimorbidity data. Clustering of the latent variables and generation of synthetic data were effective, however, both techniques need to be explored further.**

I. Introduction

This project will examine a dataset containing records of multimorbidity patients. The core research group using this data, affiliated with Newcastle University is currently undergoing the process of data treating and data preprocessing before proceeding with analysis. During such preliminary stages, and in previous literature on multimorbidity data (see section II), it has been highlighted that one of the problems of such data is its high dimensionality and the difficulty this poses for analysis such as clustering analysis. Rawson (2022) who studied the same dataset attempted to solve this problem by using topological data analysis to effectively cluster and visualize high dimensionality data. While this technique showed some promise, one limitation was that this technique was computationally expensive and therefore difficult to scale. This study proposes an alternative solution to the same problem in what should be a more efficient manner. This could be done by using autoencoders as a form of dimensionality reduction prior to clustering analysis.

The multimorbidity data that will be studied consists of 40469 patients diagnosed with 2 or more of 204 long-term conditions making this a high dimension dataset (40469 x 204). One technique for reducing dimensions of a dataset is to use an autoencoder (a type of neural network), which can encode the data from 204 dimensions to something more manageable for analysis such as 15 or 25 dimensions. However, an autoencoder does not regularize the latent space, resulting in correlated latent variables. For this reason, the performance of a variational autoencoder will be compared with the standard autoencoder which maps each latent variable onto a distribution, giving them independence and making the latent space dense. From here, further analysis can be performed such as clustering based on these 15 or 25 latent variables. As the latent variables will be densely distributed and independent, this study will also attempt to generate synthetic data like the input data.

This study will therefore begin with some background, previous literature, and a description of the data (sections II, III, IV, and V), then continue with four main pieces of work: Standard Autoencoder Analysis (section VI), Variational Autoencoder Analysis (section VII), Clustering Analysis (section VIII), and Data Generation (section IX).

II. Multimorbidity Data

Multimorbidity can be defined as 'the co-occurrence of multiple chronic or acute diseases and medical conditions within one person' (van den Akker, Buntinx, and Knottnerus, 1996). These co-occurring diseases can occur dependently or independently of each other (van den Akker, Buntinx, and Knottnerus, 1996). Consequences of multimorbidity include premature death, more frequent hospital admissions, longer hospital stays, visiting a range of medical specialists within a year (Diaz, 2021). Records show how the prevalence of multimorbidities in society is increasing and it is globally recognised that patients diagnosed with a multimorbidity are

a strain on health services (van den Akker, Buntinx, and Knottnerus, 1998). Diaz (2021) explains how these patients are high utilizers of healthcare and some of the most costly and difficult to treat patients in Europe. Studying some of the factors that may be related to multimorbidities and analysing trends of such conditions, could help in identifying risks of hospital readmissions early, and help make decisions related to the quality of care of these patients. Something of great interest to hospitals and the healthcare system (Dashtban, and Li, 2020).

Electronic recordkeeping in the healthcare system coupled with novel techniques in deep learning has recently made it possible to analyse multimorbidities more effectively, in more detail and in a more holistic way, where previously the approach has been to analyse the co-occurring diseases in isolation (Hassaine, et al. 2020). However, with such a large amount of complex data comes some challenges for analysis. One such challenge is the high dimensionality and variety of the data.

III. Autoencoders for Dimensionality Reduction

With a recent development in the ease of collecting data through electronic systems and the internet, datasets now are far richer than ever before. However, these new complex datasets can be difficult to analyse due to the high dimensionality of such data. Steinbach, Ertöz, and Kumar, (2004) discuss the 'curse of dimensionality', a phenomenon that describes how difficult datasets with large dimensions can be to analyse, which includes clustering among other analysis techniques which work well on data with lower dimensions. High dimensional data can also result in high training times for models. Zebari, et al (2020), suggests dimensionality reduction could be an effective way to manage data during pre-processing, which could reduce noise, reduce the amount of irrelevant data and redundant features, as well as solving the aforementioned challenges. One technique touched on in Zebari, et al's (2020) review is to use an autoencoder (AE) for dimensionality reduction. This technique has been shown to be more effective in reducing dimensions of multimorbidity data than other techniques such as principal component analysis (PCA) (Diaz, 2021, Cabrera-Bean, Dos Santos, Roso-Llorach, Fernández-Bertolín, Vidal, and Violán, 2020), however, Roche, Hueber, Limier, and Girin, (2018), found that a slightly more complex variational autoencoder (VAE) can reduce the data to extract more meaningful and useful variables.

This study will therefore aim to (i) see whether AE's are suitable technique for dimensionality reduction of this multimorbidity data, (ii) see whether VAE's are suitable a suitable technique for dimensionality reduction of this multimorbidity data, (iii) perform clustering analysis on encoded latent variables to gain insights of the data, and (iv) to attempt to generate synthetic data similar to that of the dataset.

IV. Autoencoders

A. Standard Autoencoders

An autoencoder is a neural network trained to encode and decode data effectively with minimal information loss. The architecture of an autoencoder usually starts with high dimensionality data and encodes the data into a representation made of fewer dimensions known as latent variables. This reduced dimension representation can be decoded to effectively reproduce the input data with minimal error. An example of the architecture can be seen in fig 1.

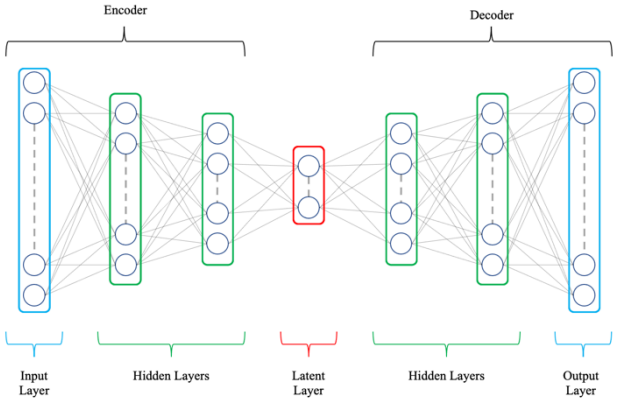


Fig. 1. Standard Autoencoder Architecture

Autoencoders are typically trained by using the same data for the input of the neural network as for the label. The loss is calculated between the output of the neural network and the label (reconstruction loss) and shows how well the output approximates the original input. The model can then be optimised by minimising the loss and can be evaluated by calculating the reconstruction loss at the end of training.

The loss that has been used to optimise the model in other studies (Diaz, 2021, Cabrera-Bean, Dos Santos, Roso-Llorach, Fernández-Bertolín, Vidal, and Violán, 2020, Roche, Hueber, Limier, and Girin, 2018) and will be used in this study is the mean square error (MSE) loss. The mean squared error is calculated by equation (1). Where n represents the number of data points, Y represents the observed values and $\hat{Y}$ represents the predicted values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \qquad (1)$$

The model can be evaluated by analysing the reconstruction loss of a training dataset and a validation dataset. To attempt to give an even representation of data in both training and validation datasets, the k-fold cross validation technique will be used. The k-fold cross validation technique trains the model k times with different training and validation datasets each time. The validation dataset is sampled without replacement from the full dataset meaning that every sample will be represented in the validation dataset once, and in the training dataset k-1 times. The average validation loss and training loss is across all training processes.

Diaz (2021) and Cabrera-Bean, Dos Santos, Roso-Llorach, Fernández-Bertolín, Vidal, and Violán, (2020) each published a study on techniques for reducing the dimensionality of multimorbidity data. Cabrera-Bean, Dos Santos, Roso-Llorach, Fernández-Bertolín, Vidal, and Violán, (2020) compared autoencoders to principal component analysis (PCA) for dimensionality reduction. They then performed clustering using K-means clustering and Expectation Maximisation modelling as a Gaussian Mixture Model. They used cluster performance metrics to analyse the effectiveness of the dimensionality reduction and found that autoencoders performed significantly better than PCAmix. This could be because PCA can only use linear

transformations to reduce dimensions whereas autoencoders can go beyond that and use non-linear transformations.

B. *Variational Autoencoders*

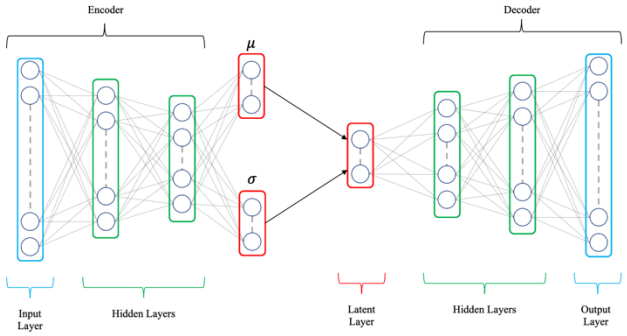


Fig. 2. Variational Autoencoder Architecture

One issue with autoencoders for dimensionality reduction is that one cannot tell whether the latent variables created in the bottleneck represent meaningful features of the data and instead, the neural network might have just learnt an efficient way of encoding the data. This can happen with autoencoders as the latent space is not regularized, meaning that any value can be chosen as a latent variable, and the latent variables can be very sparsely distributed. This kind of result would be due to accidental floating-point precision (Foundations of Data Science, 2020). One way to overcome such an issue would be to use a variational autoencoder (VAE).

While a VAE may look similar in architecture to an AE and produce similar results, one key difference is that the encoder part of the VAE reduces the input into a mean and standard deviation for each latent dimension, and then each latent variable is calculated by sampling from normal distributions with these parameters. This can be seen in Fig 2.

C. *Comparing Autoencoders and Variational Autoencoders*

Roche, Hueber, Limier, and Girin, (2018) conducted a study comparing several different types of autoencoder and their effectiveness in dimensionality reduction on sound data. One of their goals was to create independent latent variables. They compared the performance of standard autoencoders, deep autoencoders, recurrent autoencoders and variational autoencoders with principal component analysis.

They found (contrary to previous studies on the mnist dataset) that PCA outperformed standard autoencoders. They found that the best methods for reconstructing data based on reconstruction error were deep and recurrent autoencoders. VAE however, while not recording as low a reconstruction error as deep and recurrent autoencoders, they did outperform PCA in this area. The latent variables created by the VAE were also found to be most independent. This suggests that VAE produces the latent variables that represent the most meaningful features, and therefore the most useful dimensionality reduction technique for future analysis such as clustering. Methods included reporting the RMSE of each technique (For VAE, they also used a beta weighting coefficient for variations in the model), analysing correlation matrices of the latent variables for each technique.

Roche, Hueber, Limier, and Girin, (2018) showed how using a VAE cannot only achieve the same reconstruction loss as the standard AE, but the latent variables created by the VAE would be less correlated than those of a standard AE. This would suggest that these encoded latent variables represent more meaningful features of the data, and therefore could give better insight when researchers use techniques such as clustering on these latent variables.

Another benefit of training a VAE, particularly with sensitive confidential data such as health records, is that as the VAE is a probabilistic autoencoder, the decoder part of a VAE can be used as a data generator. This would allow data to be generated, to be used for analysis and to be shared with a wider research community.

V. Dataset

The dataset consists of electronic health records relating to 40469 patients diagnosed with type 1 diabetes and at least one other comorbidity of a total of 204 long-term conditions (LTC's) (including type 1 diabetes).. The amount of total LTC's diagnosed per patient ranged from 2 to 127 with a mean of 16 LTC's per patient. It should be noted that above 55 LTC's per patient was very rare with only 35 of the 40469 exceeding this amount. The event data included the date of any recorded event for each patient and LTC (events could be a diagnosis, specialist appointment, medical procedure etc.) and the data was taken from records dated between 21/10/1925 and 10/09/2021. The data also included date of birth and date of death (where applicable) of each participant and each participant is identifiable only by a patient ID.

The dataset consisted of 22055 (approx. 54.5%) Male patients and 18414 (approx. 45.5%) Female patients. The participants were approximately normally distributed across the index of multiple deprivation (IMD) quintiles with a slight negative skew, suggesting that more least deprived are diagnosed with multiple LTC's. When asked about ethnic group, 35012 (86.52%) of the participants classified themselves as White, 1891 (4.67%) as Asian or Asian British, 1540 (3.81%) as Unknown, 1161 (2.87) as Black or Black British, 574 (1.42%) as Chinese or Other Group, and 291 (0.72%) as Mixed.

For initial analysis it was decided to simplify the data into a binary format containing data only relating to whether the participant had been diagnosed with each LTC or not, with 1 denoting that the LTC was present and 0 that it was not present.

VI. Standard Autoencoder Analysis

A. *Experiments*

To create an autoencoder, one must design the architecture (as seen in Section IV, Fig.1.). This involves choosing the number of hidden layers, the dimensions of each layer, as well as nonlinear activation functions in between each layer. Other tunable parameters are the batch size that is used to train the model and the number of dimensions of the latent layer.

This study used leaky ReLU as a non-linear activation function at each layer. While the ReLu function is

the most popular activation function, leaky ReKU was chosen over the ReLU activation function as it reduces information loss at each layer. The way the ReLU activation function works is that when it is called on the output of a linear layer, any value above 0 stay as they are (y=x), however, any values 0 or below (negative values) are changed to a zero. Therefore, for any output 0 or below the additional discriminatory information is lost. The leaky ReLU function works in the same way as the standard ReLU for all positive values, however, for negative values it maps them onto a different slope to the positive values below 0. The angle of the slope is set as a parameter for the function (y=ax), and this study set it to a = 0.1. ReLU and leaky ReLU are preferred over TanH or Sigmoid activation functions as they overcome issues such as the vanishing gradient problem.

As the data was encoded into binary variables, this network is essentially working as a binary classifier (does the person have this condition or not). Therefore, the output of the final layer will be passed through a sigmoid function which has an output between 0 and 1. This is used as a probability of whether the person has the condition. Probability > 0.5 is that they have the condition and probability < 0.5 is that they do not have the condition. The output can be rounded to produce exact results of the input; however, this study does not round values within the model.

As for the hidden layers, batch size and number of dimensions, experiments were conducted on each of these in a sequence to find the optimal setting for each one. The first experiment found the optimal hidden layers, then the batch size was tested and finally the dimensions of the latent layer, selecting the optimal parameter for each experiment to be carried forward.

To train the model the k-fold cross validation technique was used (described in section IV. A..). For this analysis k=5 was chosen meaning the dataset was split 80:20 (training:validation) for each training process. The Adam optimizer was be used to train the models with a learning rate of 0.001. Each model was trained over 30 epochs.

The code implementation of the full AE for 2 and 3 hidden layers can be seen in appendix A.

(i). Hidden Layers

The first experiments tested different hidden layer structures (as visualized in Section IV, Fig.1.). The structure was symmetrical with hidden layers in the encoder and decoder of dimensions = 192-128, 192-64, 128-64, 192-128-64. These were done with a batch size of 256 and latent layer with dimensions d=15. Tab. 1. shows the training and validation loss for each architecture. The architecture with two hidden layers with dimensions 192-128 performs best and therefore will be used in further experiments.

Loss	128–64	192–64	192–128	192–128–64
Average training loss	0.000107	0.000104	0.000097	0.00011
Average validation loss	0.000109	0.000106	0.0001	0.000112

Tab. 1. Training and validation loss of autoencoder models by architecture

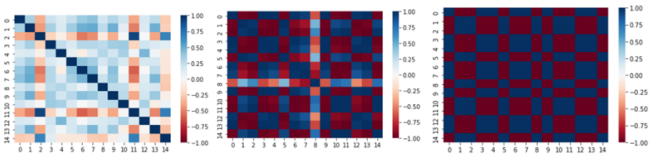
Loss	64.0	128.0	512.0	1024.0	4096.0	8192.0
AE Average training loss	0.000275	0.000149	0.000049	0.000028	0.000011	0.000006
AE Average validation loss	0.000307	0.000162	0.000050	0.000028	0.000011	0.000006

Tab. 2. Training and validation loss of autoencoder models by batch size

(ii). Batch Size

Different batch sizes were then tested with Batch size = 64, 128, 256, 512, 1024, 4096 and 8192. The loss recorded for each is displayed in the Tab. 2.

It was expected that a higher batch size would result in lower training loss but also that the model would overfit the training data. However, the validation loss continues to fall with the training loss. This unexpected result prompted further improvised investigation. For this, the correlation of the latent variables for different batch sizes was calculated and visualized (Fig 3.). This shows that while an increase in batch size may reduce loss, it produces significantly correlated latent variables, making these variables less helpful for further analysis of the data. The high correlation of latent variables at higher batch size could be due to accidental floating-point precision (outlined in Section IV. B.). This highlights that when selecting architecture, parameters and hyperparameters for dimensionality reduction for the purpose of further analysis, correlation of the latent dimensions if a critical factor to consider.



(A) Batch Size = 256 (B) Batch Size = 4096 (C) Batch Size = 8192

Fig. 3. Correlation heatmaps of autoencoder latent variables by batch size

(iii). Latent Dimensions

Several different sizes for the latent space d were tested with d = 4, 15, 25, 50 and 100. The results can be seen in Tab. 3. And visualized in Fig. 4.

Loss	4.0	15.0	25.0	50.0	100.0
AE Average training loss	0.000269	0.000147	0.000115	0.000069	0.000060
AE Average validation loss	0.000280	0.000160	0.000130	0.000084	0.000072

Tab. 3. Training and validation loss of autoencoder models by latent size

Fig. 4. shows how the loss changes depending on the value for d latent dimension. As expected, and in agreement with previous work, the loss reduces as the size of the latent space increases.

B. Discussion

This section has shown that autoencoders (AE) can be extremely effective in reducing dimensions of multimorbidity data for the purpose of data encoding with minimum information loss. However, if the purpose of

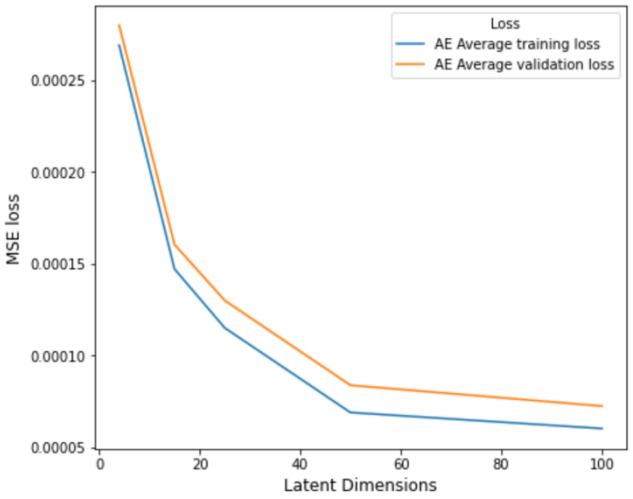


Fig. 4. MSE loss by latent space dimensions for a standard autoencoder

dimensionality reduction is for further analysis, it is important to analyze and understand the latent variables when choosing architecture, parameters and hyperparameters. A lack of this understanding could lead to many redundant, highly correlated latent variables, and consequently alter the results of further analysis such as clustering analysis.

In this section, an optimal architecture was found, these will be carried froward to the next section. This architecture has hidden layers 192-128 and takes the full form input layer – 192 – 128 – latent layer – 128 – 192 – output layer. There was no objective optimal batch size found in this section and further experiments in the next section will begin using batch size 256. The size of the latent layer will be explored again.

VII. Variational Autoencoder Analysis

The VAE (briefly outlined in section IV. B.) produces a model of the data distribution which can be express as a joint probability distribution seen in equation (2).

$$p(x,z|\theta) = p(x|z,\theta)p(z|\theta) \qquad (2)$$

Where x represents the output data, z represents the latent variables and θ represents the parameters of the decoder.

The encoder is trained to reduce the data into a distribution of latent variables that lie in the distribution $p(z,\theta)$ and can be expressed as a posterior distribution $p(z|x,\theta)$. However, this posterior distribution is intractable, therefore, the encoder must approximate the posterior distribution $q(z|x,\phi)$ (where ϕ represents the parameters of the encoder) by producing a mean and standard deviation and calculating the posterior distribution with equation (3).

$$q(z|x,\phi) = N(z;\mu_\phi(x),\sigma_\phi^2(x)) \qquad (3)$$

The model is then optimized by adjusting the parameters θ and ϕ to maximize the marginal log-likelihood $\log p(x|\theta)$, which can be written as the sum of the evidence lower bound and the KL divergence between the prior, and posterior distributions which is expressed in equation (4).

This equation also contains a β weighting term that is multiplied by the KL divergence term. This has a default value of 1 however, it can be adjusted to alter the balance of both terms.

$$log\ p(x|\theta) = KL(q(z|x,\phi)||p(z|x,\theta)) * \beta + L(q,\theta,\phi) \qquad (4)$$

Where KL represents the Kullback-Leibler divergence and $L(\phi,\theta,x)$ represents the variational lower bound. The KL divergence is then minimized when the lower bound is maximized, which can be expressed by equation (5).

$$L(\phi,\theta,x) = \mathbb{E}_q[\log p(x|z,\theta)] - KL\,(q(z|x,\phi)||p(z|\theta)) \qquad (5)$$

For this study, the KL divergence was defined as in equation (6).

$$KL = \tfrac{1}{2}\textstyle\sum_{i=1}^{k}(\sigma_\phi^2 + \mu_\phi - \log(\sigma_\phi^2) - 1) \qquad (6)$$

As the data in this study is binary, an observation $x \in \mathbb{R}^D$, given z as a multivariate Bernoulli random variable with mean μ_d. The corresponding decoder can be seen in equation (7).

$$p(x,z|\theta) = \textstyle\prod_{i=1}^{D}\mu_{di}^{xi}(1-\mu_{di})^{(1-x_i)} \qquad (7)$$

Where μ_d is the output of a neural network with parameters θ. Given this decoder, $\log p(x|z,\theta)$ is now be seen equal to the negative binary cross-entropy loss as seen in equation (8).

$$log\ p(x|z,\theta) = \sum_{i=1}^{D} x_i log\ \mu_{di} + (1 - x_i)\log(1-\mu_{di}) \qquad (8)$$

The implementation of this method within the full model can be seen in appendix B

The benefits of training a model in this way is it regularizes the latent variables by forcing them into a normal distribution. While this may increase the reconstruction loss of the model, it overcomes issues such as floating-point precision and correlated latent variables, and as the latent variables now sit in a dense distribution, samples can be taken from this distribution and decoded to produce synthetic data like the data that the model is trained on.

A. Analysis

The analysis started by building a VAE with the same architecture as the AE trained in section VI. However, the VAE proved more difficult to train, producing inconsistent reconstruction loss with an increase in latent dimensions. The loss stabilized when trained for a higher number of epochs

(50) with a decreased batch size (128). The reconstruction loss (MSE loss) across a range of dimensions (4, 10, 15, 20, 25, 30, 50) is shown in Fig. 5.

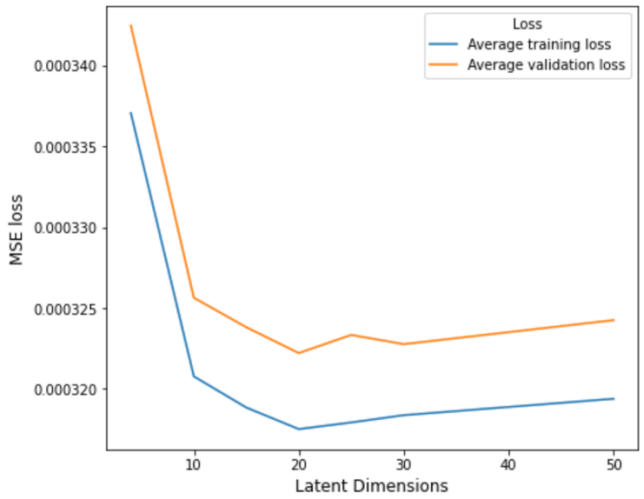


Fig. 5. MSE loss by latent space dimensions for a variational autoencoder

Fig. 5. shows that the loss reduced to begin with as dimensions increased but eventually increased again (at d=25) as the number of latent dimensions got higher. This increase could simply be due to the model not being trained for long enough, however, another explanation could be that it gets more challenging to reduce the data down to a high number of independent variables. If the latter is the case, by increasing the number of latent variables to a point where either the loss begins to rise further or the latent variables become correlated, one could find out how many independent features there were to describe the data. This could be valuable to further understanding the data, however, this is out of the scope of this study.

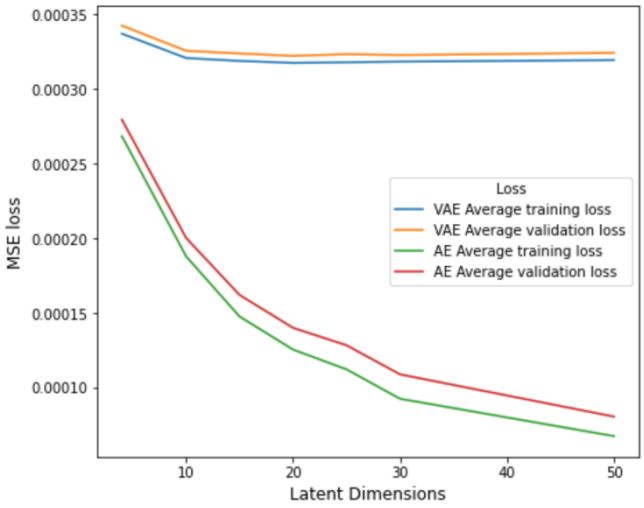


Fig. 6. MSE loss by latent space dimensions for a standard autoencoder and variational autoencoder

Fig. 6. shows the loss of the VAE model compared with an AE model trained over the same number of epochs with the same batch size. This shows that the AE dramatically outperforms the VAE with regards to reconstruction loss. However, when the relationships between the latent variables

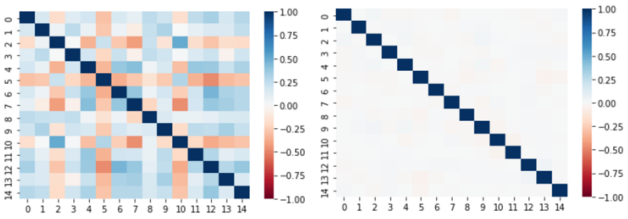


Fig. 7. Correlation of latent variables of standard autoencoder (left) and variational autoencoder (right)

of each model is shown in Fig. 7. the latent variables created by the AE are correlated while those of the VAE are independent.

Following the methods outlined by Roche, Hueber, Limier, and Girin, (2018), and outlined in section VII a beta variable was added to be multiplied by the KL-divergence part of the criterion term. This adjusts the weighting between the KL-divergence term and the reconstruction accuracy term (in this case negative cross-entropy loss). Fig. 8. shows experiments done with beta= 0.02, 0.2, 0.5 and 1.

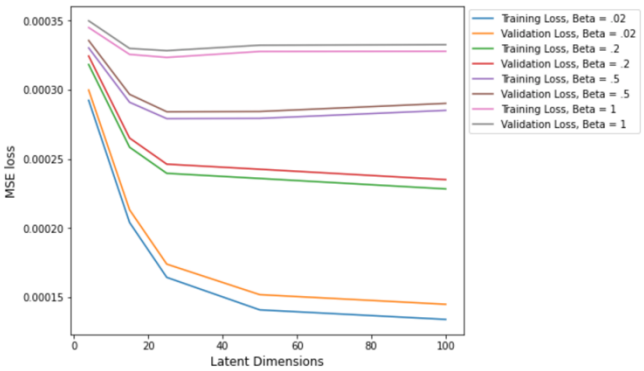


Fig. 8. MSE loss by latent space dimensions for variational autoencoders with various beta values

It can be seen here how by reducing the weight of the KL-divergence term in the criterion, also reduces the reconstruction loss of the model. However, while the loss reduces, the correlation of the latent variables increases as the value of beta decreases. This can be seen in Fig. 9. and demonstrates the trade-off between loss and latent variable independence.

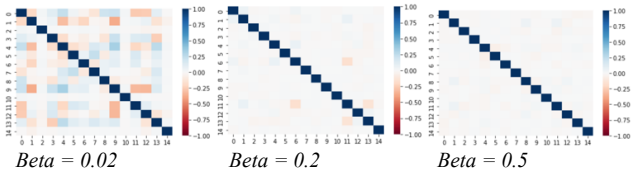


Fig. 9. Correlation of latent variables of variational autoencoders with varying beta values

B. Discussion

This section has demonstrated how variational autoencoders can effectively reduce dimensions of multimorbidity data into independent latent variables, and how changing the beta values lowers the loss but increases the correlation of the latent variables.

As a beta value of 0.2 significantly reduces the loss while not compromising significantly on latent variable independence, this will be carried forward for further analysis, with latent dimensions d = 25.

VIII. Clustering

The clustering method used in this study was k-means clustering. K-means clustering classifies data into k clusters (k selected by user). K number of points from the data are selected by the algorithm and act as initial centroids for the algorithm. The distance from each data point to each centroid is calculated and the sample is assigned to the cluster of the closest centroid. Once all samples have been assigned a cluster, a new centroid for each cluster is calculated by taking the mean of all points assigned to that cluster. This process repeats until the cluster centroids have stabilized.

The number for k can be selected by performing clustering across a variety of values for k and evaluating the performance for each k through calculating the Calinski & Harabasz score (CHS). The CHS is also known as the variance ratio criterion and draws on the ratio between the variance between clusters and the variance within clusters. Clustering that performs well would have larger variance between clusters and smaller variance within clusters, giving it a higher CHS. Once the CHS is obtained for each cluster, the best way to analyze the results and select a final value for k is to plot the results, then it can be seen if there are any significant spikes in the CHS which would indicate optimal clustering results, otherwise if the plot shows a curve, optimal clustering would be at where the curve has an 'elbow' or kink.

Before clustering was performed a VAE was trained with beta = 0.2, latent dimensions = 25, batch size = 128, and the architecture set out in section VI using the same k-fold method as described in described in section IV. A. the model with the lowest loss was then selected. This model was used to reduce dimensions of the whole dataset to 25 dimensions. K-means clustering was performed with k ranging from 2-20. The CHS was calculated, the results were plotted and can be seen in Fig. 9.

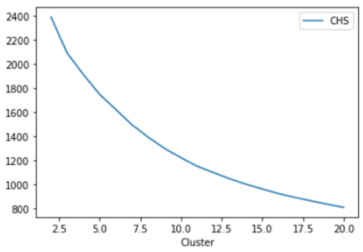


Fig. 9. CHS by number of clusters

As there is no obvious elbow in Fig. 9. There is no obvious optimal number of clusters, therefore, for simplicity to test the concept of clustering with these latent variables k=3 will be chosen. Principal component analysis (PCA) was then used as a practical method to reduce the dimensions of the data to two variables for the purpose of visualization. The results of the clustering analysis can be seen in Fig. 10.

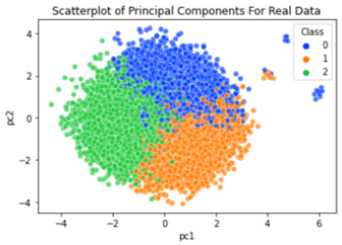


Fig. 10. Number of clusters for principal components 1 and 2 of real data

To attempt to gain a better understanding of the clusters at k=3, the prevalence of each of the LTC's was taken for each cluster and compared to one another (diabetes type 1 was removed from the data as this was the criteria for including the data in the study). The full table of the 10 most prevalent LTC's and they're prevalence percentage from each cluster can be seen in appendix C.

The analysis found that there were some common factors across all clusters namely Type 2 Diabetes, Unspecified or Rare Diabetes, Hypertension, Painful conditions, and Diabetic eye disease. This is unsurprising as the dataset is based on people who have Type 1 Diabetes.

Beyond this there are clear distinguishing features of the clusters. C0 patients have particularly high prevalence of mental health conditions such as depression and anxiety, C2 patients have particularly high prevalence of conditions related to the digestive system such as gastro-oesophageal reflux disease, gastritis and duodenitis, and oesophagitis and oesophageal ulcers, while C1 does not seem to have any conditions with particularly high prevalence other than those identified as common to all clusters.

IX. Synthetic Data Generator

For synthetic data, the same model outlined in used for clustering in section VII will be used for dimensionality reduction. The full binary dataset was passed through the VAE, reducing the dimensions of the data to 25 as with the clustering analysis. The means and standard deviations of these 25 latent variables were then used as parameters of a normal distribution. 40469 samples were drawn from these distributions and decoded. The output was then rounded (to either 1 or 0) to create the final synthetic dataset. To give a fair comparison between the real and synthetic data, the real data was encoded and decoded, and the reconstruction of the data was used for comparison. This was instead of using the original full dataset to account for reconstruction loss.

	Generated Rows	Real Rows
count	40469.000000	40469.000000
mean	11.974746	11.932985
std	7.239506	7.125338
min	1.000000	1.000000
25%	7.000000	7.000000
50%	10.000000	11.000000
75%	16.000000	16.000000
max	76.000000	125.000000

Tab. 4. Comparison of total number of conditions per person between real and synthetic data.

For initial comparison between reconstructed and synthetic data, the number of LTC's each person was diagnosed with was calculated for both datasets and the descriptive statistics compared. Tab. 4 shows how, with the exception to the maximum number of conditions that a participant is diagnosed with, the descriptive statistics are extremely similar. The biggest exception (max number of LTCs diagnosed) is likely due to the generated data not generating some of the rarer cases of a very large number of LTC's. Next the prevalence of each condition was calculated as a percentage. The difference between the prevalence of LTC's in the real and synthetic data was calculated and the mean difference across all conditions was 0.00875% (3 significant figures).

Clusters were predicted for the synthetic data based on the k-means model trained in section VIII. The data was distributed in similar proportions to the original data as can be seen in Tab. 5. The visualization of how the data was classified can be seen in Fig. 11.

	Cluster 0	Cluster 1	Cluster 2
Real Proportions	0.336776	0.327115	0.336109
Synthetic Proportions	0.338704	0.339173	0.322123

Tab. 5. Proportion of data per cluster for real and synthetic data

The prevalence of diseases for each predicted class among the synthetic data can be seen in appendix D. Cluster 2 of the synthetic data had the closest resemblance to the same cluster of the real data with LTCs relating to the digestive system being prevalent in this cluster also. Other than this there are differences between the clusters of the real and synthetic data. This could be due to 3 being too small a number for clustering of this data. Future work should compare these predictions of synthetic data with predictions of unseen real data. To gain better insight, future work should exclude all common conditions across the clusters, although, once the clusters are established, the proportions of these common conditions could be important to understand relationships between conditions.

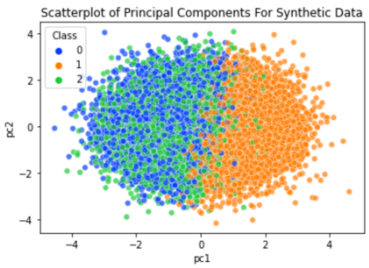


Fig. 11. Predicted classes for principal components 1 and 2 of synthetic data

X. Discussion

This work has successfully demonstrated how standard autoencoders (AE) and variational autoencoders (VAE) can be used effectively in reducing dimensions of multimorbidity data. It found that AE returned the lowest loss and would work well for data encoding for minimizing storage space. VAE was the preferred method of dimensionality reduction for future analysis due to the independence of latent variables. A VAE was used to reduce dimensions to cluster data effectively and each cluster showed distinguishing features. Synthetic data was successfully created with remarkable similarity to the real data. The synthetic data was then clusters based on the clustering model created in the previous section.

There are, however, some limitations to this study. The ad hoc nature that some aspects of the study took on due to unexpected results should be improved by systematic exhaustive experiments in future. Possibly taking advantage of techniques such as a grid search. This would give a more in-depth understanding of the data and techniques used. Another limitation is that the data was simplified to a binary dataset losing details of the dataset. Using age at date of diagnosis could be an avenue for future research which might capture richer latent representations of the data. A final limitation was that only 3 clusters were specified, and while the results show they were distinguishable from one another, future research should aim to create more clusters as there are undoubtedly more than 3 in this dataset.

Future study could also understand the latent variables better by minimizing the standard deviation of each of all but one of the latent variables and keeping the other variable with its full range. Data could be sampled from these distributions and any differences in the reconstruction data could be attributed to the one latent variable that has kept its full range of values. Doing this for all latent variables would give insight into understanding the latent variables and analysis conducted with them.

XI. Conclusion

This study has shown how AEs and VAEs can effectively reduce dimensions of multimorbidity data. Following dimensionality reduction using a VAE, clustering was successfully preformed, and synthetic data successfully generated. Future research should look at deeper analysis in clustering and synthetic data to understand the full capabilities of these techniques with regards to latent variables of this dataset.

XII. References

Academy of medical sciences (Royaume uni)., 2018. Multimorbidity: a priority for global health research. Academy of medical sciences.

Cabrera-Bean, M., Dos Santos, V.J.P., Roso-Llorach, A., Fernández-Bertolín, S., Vidal, J. and Violán, C., 2020, July. Autoencoders for health improvement by compressing the set of patient features. In 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC) (pp. 5917-5920). IEEE.

Chushig-Muzo, D., Soguero-Ruiz, C., de Miguel-Bohoyo, P. and Mora-Jiménez, I., 2021. Interpreting clinical latent representations using autoencoders and probabilistic models. Artificial Intelligence in Medicine, 122, p.102211.

Dambha-Miller, H., Simpson, G., Akyea, R.K., Hounkpatin, H., Morrison, L., Gibson, J., Stokes, J., Islam, N., Chapman, A., Stuart, B. and Zaccardi, F., 2022. Development and Validation of Population Clusters for Integrating Health and Social Care: Protocol for a Mixed Methods Study in Multiple Long-Term Conditions (Cluster-Artificial Intelligence for Multiple Long-Term Conditions). JMIR research protocols, 11(6), p.e34405.

Dashtban, M. and Li, W.V., 2020. Predicting risk of hospital readmission for comorbidity patients through a novel deep learning framework.

Díaz, J. A., 2021. Longitudinal analysis of multimorbidity patterns in people over 65 years (Bachelor's thesis, Universitat Politècnica de Catalunya).

Foundations of Data Science (2020) Autoencoders In Practice. Available at: https://www.youtube.com/watch?v=viOjfvP7Fqc&list=PLknxdt7zG11PgvbnOEsTKi_EJM4Pn-jsU&index=7&t=1361s.

Hassaine, A., Canoy, D., Solares, J.R.A., Zhu, Y., Rao, S., Li, Y., Zottoli, M., Rahimi, K. and Salimi-Khorshidi, G., 2020. Learning multimorbidity patterns from electronic health records using non-negative matrix factorisation. *Journal of Biomedical Informatics*, *112*, p.103606.

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

Rawson, J., 2022. Using Topological Data Analysis (TDA) to Dissect Multimorbidity in the East London Patient Population.

Roche, F., Hueber, T., Limier, S. and Girin, L., 2018. Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent and variational models. arXiv preprint arXiv:1806.04096.

Steinbach, M., Ertöz, L. and Kumar, V., 2004. The challenges of clustering high dimensional data. In New directions in statistical physics (pp. 273-309). Springer, Berlin, Heidelberg.

van den Akker, M., Buntinx, F. and Knottnerus, J.A., 1996. Comorbidity or multimorbidity: what's in a name? A review of literature. The European journal of general practice, 2(2), pp.65-70.

Van den Akker, M., Buntinx, F., Metsemakers, J.F., Roos, S. and Knottnerus, J.A., 1998. Multimorbidity in general practice: prevalence, incidence, and determinants of co-occurring chronic and recurrent diseases. Journal of clinical epidemiology, 51(5), pp.367-375.

van den Akker, M., Buntinx, F. and Knottnerus, J.A., 1996. Comorbidity or multimorbidity: what's in a name? A review of literature. The European journal of general practice, 2(2), pp.65-70.

Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D. and Saeed, J., 2020. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. Journal of Applied Science and Technology Trends, 1(2), pp.56-70.

Appendices

Appendix A

Autoencoders

Below are the classes for the autoencoders with both 2 and 3 hidden layers. The input size, dimensions of the hidden layers, the latent size are the tunable parameters of the models.

Each class contains 3 functions:

The encode function - which reduces input data to the dimensions of the specified latent size

The decode function - which decodes data from latent variable representation back into data with original dimensions.

The forward function - which is a standard function in neural networks which when called, in this case returns the output from the whole process of encoding and decoding data.

```
class AE2(nn.Module):
    def __init__(self, input_size, hidden_size1, hidden_size2, latent_size):
        super().__init__()
        self.fc1 = nn.Linear(input_size, hidden_size1)
        self.fc2 = nn.Linear(hidden_size1, hidden_size2)
        self.fc3 = nn.Linear(hidden_size2, latent_size)
        self.fc4 = nn.Linear(latent_size, hidden_size2)
        self.fc5 = nn.Linear(hidden_size2, hidden_size1)
        self.fc6 = nn.Linear(hidden_size1, input_size)
        self.lrelu = nn.LeakyReLU(0.1)

    def encode(self, x):
        p_x = self.lrelu(self.fc1(x))
        p_x = self.lrelu(self.fc2(p_x))
        p_x = self.lrelu(self.fc3(p_x))

        return p_x

    def decode(self, z_x):
        q_x = self.lrelu(self.fc4(z_x))
        q_x = self.lrelu(self.fc5(q_x))
        q_x = torch.sigmoid(self.fc6(q_x))

        return q_x

    def forward(self, x):
        p_x = self.encode(x)
        q_z = self.decode(p_x)

        return q_z, p_x
```

```
class AE3(nn.Module):
    def __init__(self, input_size, hidden_size1, hidden_size2, latent_size):
        super().__init__()
        self.fc1 = nn.Linear(input_size, hidden_size1)
        self.fc2 = nn.Linear(hidden_size1, hidden_size2)
        self.fc3 = nn.Linear(hidden_size2, hidden_size3)
        self.fc4 = nn.Linear(hidden_size3, latent_size)
        self.fc5 = nn.Linear(latent_size, hidden_size3)
        self.fc6 = nn.Linear(hidden_size3, hidden_size2)
        self.fc7 = nn.Linear(hidden_size2, hidden_size1)
        self.fc8 = nn.Linear(hidden_size1, input_size)
        self.lrelu = nn.LeakyReLU(0.1)

    def encode(self, x):
        p_x = self.lrelu(self.fc1(x))
        p_x = self.lrelu(self.fc2(p_x))
        p_x = self.lrelu(self.fc3(p_x))

        return p_x

    def decode(self, z_x):
        q_x = self.lrelu(self.fc4(z_x))
        q_x = self.lrelu(self.fc5(q_x))
        q_x = torch.sigmoid(self.fc6(q_x))

        return q_x

    def forward(self, x):
        p_x = self.encode(x)
        q_z = self.decode(p_x)

        return q_z, p_x
```

Appendix B

Variational Autoencoder

Below is the class for the variational autoencoder with 2 hidden layers. The input size, dimensions of the hidden layers, the latent size are the tunable parameters of the models.

The class contains the same 3 functions as the standard AE although they work slightly differently and produce different outputs:

The encode function - this reduced the data to a mean and standard deviation of the latent vairables. The latent variables are then obtained by sampling from normal distributions with the means and standard deviations (which is multiplied by some noise or error taken from a normal distribution) of the latent variables. The encode function takes data as an input and produces the latent variables, mean, log variance, and standard deviation as the ouptut.

The decode function - takes latent variables as an input and decodes the data from latent variable representation back into data with original dimensions.

The forward function - in this case the forward function takes the data and the beta weighting value as input. It runs the data through the encode and decode functions, and returns the optimization criterion (made up of the sum of the kl divergence term multiplied by beta, and negative cross entropy loss), the ouput of the decode function and the latent variables.

```
class BernoulliVAE2(nn.Module):
    def __init__(self, input_size, hidden_size1, hidden_size2,  latent_size):
        super(BernoulliVAE2, self).__init__()
        # Encoder parameters
        self.fc1 = nn.Linear(input_size, hidden_size1)
        self.fc2 = nn.Linear(hidden_size1, hidden_size2)
        self.fc3 = nn.Linear(hidden_size2, latent_size)
        self.lrelu = nn.LeakyReLU(0.1)
        self.enc_mu = nn.Linear(latent_size, latent_size)
        self.enc_logvar = nn.Linear(latent_size, latent_size)

        # Distribution to sample for the reparameterization trick
        self.normal_dist = MultivariateNormal(torch.zeros(latent_size),
                                              torch.eye(latent_size))

        # Decoder parameters
        self.fc4 = nn.Linear(latent_size, hidden_size2)
        self.fc5 = nn.Linear(hidden_size2, hidden_size1)
        self.fc6 = nn.Linear(hidden_size1, input_size)









        # Reconstruction loss: binary cross-entropy
        self.criterion = nn.BCELoss(reduction='sum')
```

```
    def encode(self, x):
        # Obtain the parameters of the latent variable distribution
        h = self.lrelu(self.fc1(x))
        h = self.lrelu(self.fc2(h))
        h = self.lrelu(self.fc3(h))
        mu_e = self.enc_mu(h)
        logvar_e = self.enc_logvar(h)
        std = torch.exp(0.5 * logvar_e)
        noise = torch.randn_like(std)

        # Get a latent variable sample with the reparameterization trick

        z = mu_e + (std * noise)

        return z, mu_e, logvar_e, std

    def decode(self, z):
        # Obtain the parameters of the observation distribution
        h = self.lrelu(self.fc4(z))
        h = self.lrelu(self.fc5(h))
        output = torch.sigmoid(self.fc6(h))

        return output
```

```
    def forward(self, x, beta):
        """ Calculate the negative lower bound for the given input """
        z, mu_e, logvar_e, std = self.encode(x)
        output = self.decode(z)
        neg_cross_entropy = self.criterion(output, x)
        kl_div = -0.5* (1 + logvar_e - mu_e**2 - torch.exp(logvar_e)).sum()
        beta = beta

        # Since the optimizer minimizes, we return the negative
        # of the lower bound that we need to maximize
        return neg_cross_entropy + kl_div*beta, output, z
```

Appendix C

The table contains disease prevalence within clusters created from real data.

Rank	Cluster 0	Cluster 1	Cluster 2
1.	**Depression (94.41%)**	Unspecified or Rare Diabetes (79.12%)	Type 2 Diabetes (85.10%)
2.	Unspecified or Rare Diabetes (80.88%)	Type 2 Diabetes (62.22%)	Unspecified or Rare Diabetes (82.84%)
3.	Type 2 Diabetes (73.66%)	Hypertension (55.95%)	Hypertension (80.72%)
4.	Hypertension (66.11%)	Diabetic eye disease (53.57%)	**Gastro-oesophageal reflux disease 59.08%)**
5.	Painful conditions (53.07%)	Painful conditions (29.61%)	Painful conditions (57.95%)
6.	Diabetic eye disease (52.04%)	Enthesopathies & synovial disorders (28.96%)	Diabetic eye disease (56.63%)
7.	**Anxiety and phobia (46.36%)**	Coronary heart disease (25.67%)	**Gastritis and duodenitis (50.43%)**
8.	Constipation (39.14%)	**Dermatitis (24.51%)**	**Oesophagitis and oesophageal ulcer (47.22%)**
9.	Enthesopathies & synovial disorders (38.88%)	Chronic Kidney Disease (23.47%)	Coronary heart disease (46.79%)
10.	Coronary heart disease (35.97%)	Constipation (22.20%)	Chronic Kidney Disease (46.17%)

Appendix D

The table contains disease prevalence within clusters assigned to synthetic data, based on clusters created from real data.

Rank	Cluster 0	Cluster 1	Cluster 2
1.	Type 2 Diabetes (93.988%)	Unspecified or Rare Diabetes (82.76%)	Unspecified or Rare Diabetes (81.55%)
2.	Hypertension (88.74%)	Diabetic eye disease (52.06%)	Type 2 Diabetes (79.26%)
3.	Unspecified or Rare Diabetes (83.29%)	Type 2 Diabetes (40.48%)	Hypertension (74.58%)
4.	Painful conditions (70.67%)	Hypertension (37.42%)	Painful conditions (58.85%)
5.	Coronary heart disease (63.00%)	**Depression (34.21%)**	**Gastritis and duodenitis (58.52%)**
6.	Diabetic eye disease (57.11%)	Enthesopathies & synovial disorders (27.96%)	Diabetic eye disease (56.94%)
7.	**Chronic Kidney Disease (52.25%)**	Erectile dysfunction (18.69%)	Depression (49.11%)
8.	**Osteoarthritis (excl spine) (50.88%)**	**Dermatitis (16.12%)**	**Gastro-oesophageal reflux disease (48.46%)**
9.	**Other anaemias (49.78%)**	Asthma (15.52%)	Coronary heart disease (45.85%)
10.	**Constipation (48.53%)**	Painful conditions (14.59%)	Enthesopathies & synovial disorders (43.76%)