Final Year Project Report

Imperial College London

Department of Electrical and Electronic Engineering

Patient similarity retrieval based on laboratory data cluster analysis and visualisation

Student: **Oliver Stiff**

CID: **01352628**

Supervisor: **Dr Bernard Hernandez Perez**

Second marker: **Dr Timothy Constandinou**

Final Report Plagiarism Statement

I affirm that I have submitted, or will submit, an electronic copy of my final year project report to the provided EEE link.

I affirm that I have provided explicit references for all the material in my Final Report that is not authored by me, but is represented as my own work.

Acknowledgements

I would like to extend my deepest gratitude to my supervisor, Dr Bernard Hernandez Perez, for his unwavering support and guidance throughout this project. I would like to thank him for the opportunity to work on this topic and the invaluable insights and knowledge he provided.

I also cannot begin to express my thanks to my parents and sisters for their unconditional support and encouragement, without which I would not be where I am today.

Abstract

Large scale medical datasets and electronic health records have become increasingly common, prompting the creation of clinical decision support systems which utilise data to enhance health care. Tools have been designed to predict the likelihood of infection, automate drug dosing and prescriptions, and evaluate a patient's risk of death. Adoption, however, remains low due to inherent issues, most notably, concern from clinicians regarding the quality of recommendations and lack of transparency about how results are obtained.

This report covers the design and implementation of a system that uses laboratory test data to retrieve and present information relevant to a current patient, allowing clinicians to make quicker diagnoses and create tailored treatment plans based on past data. Thus, using the increasing amount of healthcare data without impinging on clinicians' diagnosis and decision-making process and avoiding the typical objections to automated systems.

The proposed solution incorporates the results of dimensionality reduction and clustering algorithms into an interactive, web-based similarity retrieval application, demonstrating the potential of unsupervised-learning based tools in clinical settings.

Contents

List of Tables

List of Figures

Acronyms

AE	Autoencoder
CBR	Cased-Based Reasoning
CDSS	Clinical Decision Support System
CSS	Cascading Style Sheets
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DR	Dimensionality Reduction
EHR	Electronic Health Records
GMM	Gaussian Mixture Model
GUI	Graphical User Interface
HCT	Haematocrit
HTML	Hypertext Markup Language
IQR	Inter Quartile Range
JS	JavaScript
JSON	JavaScript Object Notation
LDA	Linear Discriminant Analysis
NN	Neural Network
PCA	Principal Component Analysis
PLT	Platelets
REST	Representational State Transfer
SOM	Self-Organising Map
t-SNE	t-distributed Stochastic Neighbour Embedding

Chapter 1

Introduction

1.1 Motivation

With electronic health records (EHR) having become commonplace in many countries [1, 2], a wealth of information is being recorded for every patient, creating large sets of medical data. This comes as a replacement to traditional paper-based records which aside from consuming an increasing amount of space, can impact access to timely medical care [3]. Information ranging from basic demographic data such as age and gender, administration data such as date of admission and discharge, and laboratory test results and treatment plans are increasingly being recorded electronically [4, 5]. This gradually increasing availability of data opens the door to some advanced clinical decision support systems (CDSSs) which can aid clinicians in making faster, better informed decisions [6].

The growing adoption of electronic health records has led to the creation of systems that provide clinicians with electronic alerts and reminders, patient-specific diagnostics, and automated treatment recommendations [7, 8]. Specific tools have been designed to predict the likelihood of infection [9], automate drug dosing and prescriptions [10, 11] and evaluate a patient's risk of death [12] or disease severity [13]. Whilst some systems have demonstrated the ability to improve the quality of care [14, 15], the adoption of clinical decision support systems remains low due to inherent issues with EHRs and the automation of healthcare decisions [16]. EHR platforms and CDSSs are typically implemented independently, leading to incompatibilities and inconsistencies in how data is recorded [17]. Furthermore, the black-box approach taken by these systems has induced concern from clinicians regarding the quality of recommendations [18, 19] and lack of clarity about how results are obtained [20].

Clinicians typically rely on existing knowledge when making decisions and consider previous patients when performing a diagnosis or formulating a treatment plan [21, 22]. Thus, this project aims to use laboratory test data to cluster patients, creating visualisations that can be used to inspect similar patients, their symptoms, test results and outcomes, facilitating the re-use of previously obtained data.

1.2 Project definition

Performing patient similarity retrieval and detecting patterns in laboratory results is the focus of this report, with the hope that this can help healthcare professionals make clinical decisions. Several obstacles, however, stand in the way of the efficient discovery of patterns in medical data. The first is the dimensionality of the data. There are hundreds of potential biochemical markers in medical records which can make training models highly computationally intensive. The high dimensionality causes a second issue, which is data sparsity. Most patients will only have a select few laboratory tests making it impossible to compare them to patients who have had different tests. Finally, the sparse, high-dimensional data is more likely to be impacted by outliers causing unreliable results.

This report aims to identify dimensionality reduction algorithms best suited to similarity retrieval and visualisation of medical data. That is, algorithms that can create meaningful, low dimensional representations of high dimensional data. These low-dimensional representations of the data will subsequently be analysed using unsupervised clustering algorithms and incorporated into a system that can identify patients deemed most similar to a given patient.

The aforementioned research will be integrated into an interactive web-based application, creating an interface to the patient case base that facilitates retrieving information relevant to a current patient's treatment using patient similarity retrieval.

1.3 Report structure

The report is organised as follows. Chapter 2 provides the background material relevant to the motivation and technical aspects of the project, including data-mining and unsupervised learning concepts. The project requirements are outlined in Chapter 3, along with the proposed system architecture and design. Chapter 4 presents key implementation decisions relating to the web application and the algorithms used to perform dimensionality reduction, clustering and similarity retrieval. The benefits and drawbacks of various algorithms are analysed in Chapter 5 through a series of experiments to compare performance when applied to a medical dataset. The chapter also evaluates the web application's performance through a series of tests and metrics. Chapter 6 provides an analysis of the implemented system and a critical evaluation of the work performed, revisiting the system requirements. Closing remarks and potential further work are outlined in Chapter 7. Finally, a user guide is presented in Chapter 8, detailing the system setup process and describing the user interaction process with the graphical user interface.

Chapter 2

Background

This chapter provides an overview of the concepts surrounding patient similarity retrieval, such as medical data (subsection 2.1.1), case-based reasoning and clinical decision support systems (subsection 2.1.2), and the application of clustering to clinical data (subsection 2.1.3). In addition, the technical concepts relating to the implementation of a similarity retrieval system and the visualisation of medical data are presented in section 2.2. These include data mining (subsection 2.2.1), dimensionality reduction (subsection 2.2.5) and clustering (subsection 2.2.6).

2.1 Data-oriented clinical decision methods

Data plays an essential role when making decisions in a clinical setting or performing a medical diagnosis. Utilising a patient's medical history, demographic knowledge, and records of past patients can accelerate and improve decision-making, resulting in better treatment plans and more timely administration of medical care.

2.1.1 Medical data and electronic health records

The adoption of electronic health records and electronic medical records—terms often used interchangeably—has led to an increase in the availability and quantity of medical data in an electronic format. In addition, large datasets have been obtained as the result of studies and research into specific diseases or illnesses.

The increasing amount of data being recorded in an electronic format, be it in electronic health records or standalone datasets, contributes to an increasingly large knowledge base. The information being recorded is extensive and includes several different categories. Records often include meta-data relating to how, when and why the information was catalogued, patient demographics such as age and gender, and information directly relating to diagnosis and treatment: symptoms, test results, therapies and outcomes being representative examples [4, 5].

This data can then be used in research, the development of automated systems and even directly by clinicians. Diagnoses and the preparation of treatment plans will often be partially based on previously obtained information. Details such as which treatment or dosing of a drug performed best for a given patient demographic can be invaluable information, helping experienced clinicians and, more notably, junior clinicians who do not yet have a vast number of past cases or experiences to base themselves on.

2.1.2 Case-based reasoning and CDSSs

The concept of diagnosing and treating a new patient based on past patient data can be likened to case-based reasoning (CBR). Indeed, CBR is a process by which a problem is solved based on solutions to similar problems encountered in the past. This concept has been described as the combination of four processes [24]: retrieve, reuse, revise and retain.

The first stage, "retrieve", identifies several cases similar to the current query or the current patient in the case of medical data. These cases and associated information are retrieved from the knowledge base. The second step, "reuse", applies the solutions from previous problems to the current one. This can mean adapting or combining several solutions or treatment plans to solve the current problem appropriately. The "revise" stage of CBR analyses the solution identified in the previous step to determine if the adaptations were beneficial or detrimental. Identifying which adaptations or combinations are more reliable or likely to result in a successful outcome can improve the case-based reasoning process. Finally, the "retain" portion of the cycle adds the current case and solution to the case base if it is deemed likely to be beneficial in the future.

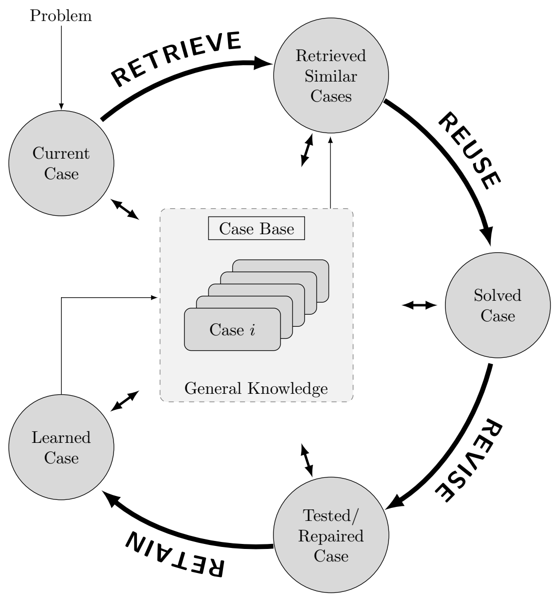


Figure 2.1: Case-based reasoning cycle [23]. Diagram illustrating the four stages of case-based reasoning: retrieve, reuse, revise and retain.

As demonstrated in Figure 2.1, the case-based reasoning cycle is supported at all stages by previously obtained knowledge. This knowledge can include: how to retrieve the most relevant information, how to analyse it and extract valuable components appropriately, and how to adapt a solution and determine which parts of it are likely to be helpful in the future.

Clinical decision support systems

Clinical decision support systems are software-based systems that can perform various tasks, primarily oriented around assisting healthcare professionals in making decisions. These systems can be implemented in various ways, relying on user-defined rules or machine learning models. For example, some systems implement parts of the case-based reasoning cycle, relying on EHRs as a knowledge base. Examples of the tasks performed by CDSSs include automated treatment recommendations, the dispatch of alerts and warnings to clinicians [7, 8], drug dosing suggestions [10, 11] and determining the likelihoods of infection [9] or death [12]. In addition, while few systems implement the complete CBR cycle, there are many examples of systems that perform the "reuse" stage, producing recommended treatments or outputting numbers such as probabilities corresponding to specific outcomes. However, lack of clarity on the methods utilised to obtain these results or the inner workings of models learned through machine-learning techniques, along with other concerns, has impacted the adoption rate of CDSSs [16, 18, 19, 20].

Thus, this project focuses on the "retrieval" stage of case-based reasoning, identifying past cases most similar to a present case. In doing so, the project aims to produce a system that facilitates and improves the information retrieval process, creating useful visualisations and statistics whilst mitigating the aforementioned concerns and allowing clinicians to form their own judgements and treatment plans based on the retrieved information. In addition, a better understanding of the data can also be obtained by applying clustering techniques to identify patterns or subpopulations in the data.

2.1.3 Clustering medical data

Clustering has been applied to medical datasets in an attempt to solve various issues or identify trends and patterns in the data. This has been done with varying degrees of success, with some methods failing altogether. However, several studies have successfully shown that unsupervised learning methods, and more specifically clustering, can be applied to medical data to identify patterns or groups in a set of patients.

Identifying disease sub-types

One of the most common applications of clustering on medical data is the use of electronic health records or raw laboratory results to identify subgroups in patients who all have the same disease or condition. Clusters have been identified in patients with hypertension and diabetes [25], patients with IgG4-Related Disease [26] and patients who underwent joint arthroplasty [27] amongst others. These studies have shown that a wide variety of clustering algorithms are available, and which one is most appropriate is highly dependent on the data being used. K-means is widely used due to its simplicity but more versatile algorithms like DBSCAN are sometimes necessary when searching for more complex clusters (subsection 2.2.6). These studies also reveal that additional steps are often needed before clustering to obtain meaningful results. Preprocessing (subsection 2.2.4) is a ubiquitous stage in data mining, with some methods leading to better performance, regardless of which clustering algorithm is used [28]. Dimensionality reduction methods are also commonplace for various reasons. They have been used to make the clustering phase more effective by reducing the number of features present in the dataset [26, 29]. They have also been used due to their ability to make data easier to visualise. Reducing the dimension of a dataset through techniques like self-organising maps can make visualising high-dimensional data in a low dimensional space possible. This makes identifying patterns visually substantially easier [25].

Outlier detection

Clustering has also been applied to health records and other types of medical data to identify outliers. This can be used to detect errors or implausible values in EHRs [30]. Results showed that an unsupervised learning approach can yield better performance than a conventional rule-based system. Another study showed that outlier detection could potentially be used to identify acute coronary syndrome patients who were at higher risk [31], further demonstrating the potential strengths of unsupervised learning.

Applying clustering to more general problems

Whilst most of the clustering on medical data has been focused on specific diseases or conditions, there have been attempts to use less specialised datasets. One study clustered patients using electronic health records without filtering for a specific disease, the aim being to help healthcare professionals diagnose and treat patients who were categorised together and, therefore, likely to have similar conditions [32]. There have also been attempts to use laboratory data such as biomarkers to cluster patients [28]. Whilst these tests were more effective when looking at patients affected by a same condition (diabetes), there are indications that valuable information can be gained from this type of clustering.

Unsuccessful applications of clustering

Whilst clustering has been applied to medical data with impressive results; there are cases where this has been unsuccessful. One study attempted to use clustering methods to detect critical acute coronary syndrome patients [31]. No clusters were formed, showing that not all data and tasks are suited to unsupervised learning applications and demonstrating why supervised learning is the method of choice when the data allows it.

The choice of which clustering, preprocessing, and dimensionality reduction algorithms are used are all essential to obtain reliable results. Furthermore, failing to account for noise or how an algorithm will affect the data can severely impact results [33].

2.2 Background theory

This section introduces the fundamental technical notions relevant to the implementation and analysis presented in this report. These include data mining, dimensionality reduction and clustering concepts.

2.2.1 Knowledge discovery from data

Knowledge discovery from data (KDD), often called data mining, is a process used to discover interesting patterns in data and make sense of that data [34]. A better understanding of data can reveal knowledge about the domain it describes and allow for novel solutions to existing questions and problems. *Data Mining: concepts and techniques* summarises the knowledge discovery process into 7 stages as follows [35]:

1. Data cleaning
2. Data integration
3. Data selection
4. Data transformation
5. Data mining
6. Pattern evaluation
7. Knowledge representation

Data cleaning, integration, selection and transformation are all forms of data preprocessing (subsection 2.2.4). The data mining step refers to the usage of intelligent methods, such as unsupervised learning, to extract patterns from the dataset (subsection 2.2.2). Pattern evaluation is the act of extracting patterns which are deemed "interesting", that is, patterns which can be easily understood, reliably applied to new data, and have a potential use case. Knowledge representation deals with the visualisation of the extracted patterns and data in a way which is easy to analyse and make use of.

2.2.2 Unsupervised learning

Unsupervised learning (UL) is a machine learning method that aims to uncover significant patterns in data and understand underlying structures in datasets [36, 37]. It does not rely on pre-existing labels or human supervision and is, therefore, often called self-organisation or "learning without a teacher" [38, 39]. There are many approaches to unsupervised learning, with the most common including clustering, neural networks, and anomaly detection.

In unsupervised learning applications, the input data dimension is often higher than what would be seen in supervised learning, and the properties being extracted more complex. Therefore, many UL methods focus on identifying relations between variables and finding models that can represent the data instead of predicting an output variable based on a set of input variables.

2.2.3 Supervised learning

Supervised learning (SL), like unsupervised learning, is a method that aims to uncover underlying structure in an input dataset. However, unlike unsupervised learning, it does so to learn a function that maps input data to one or more outputs. This function is learnt during training by observing example input-output pairs. Given a set of data with N observations, $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where x_i is an input and y_i is its associated output as generated by an unknown function $y = f(x)$, a supervised learning algorithm aims to uncover a function h which best approximates f. The performance of such a model is evaluated based upon its ability to generalise and perform well when presented with previously unseen data. [40]

2.2.4 Preprocessing

Preprocessing is a vital step in the KDD process because many data mining algorithms have difficulty handling large scale, high-dimensional and sparse datasets, all of which are common in medical data [41, 42, 43]. Data preparation addresses the aforementioned issues, as well as biases in data and incomplete or inconsistent data [44, 45]. There are four commonly used stages when preparing a dataset for data mining: (i) data cleaning, (ii) feature scaling, (iii) feature engineering, and (iv) imbalanced data handling [45]. These steps, similar to the first four stages of the KDD process presented in subsection 2.2.1, are described below.

Data cleaning

Data cleaning or cleansing is the process of discovering and modifying erroneous entries in a dataset. Modifying could mean, inputting new data, correcting existing data or removing tuples altogether. Tuples with missing attribute values can either be removed from the dataset or have the missing attribute values inputted based on an estimated value. If data imputation is used, the dataset needs to be analysed to determine what proportion of the data is missing and if missing entries are linked to another attribute. This analysis influences which imputation method is used, with the attribute's mean or median being common choices [41, 45]. Inconsistencies in data such as varying formats in dates or codes need to be corrected [46]. Finally, noisy data which may be caused by both errors and outliers needs to be handled. This can be done by removing entries which fall outside a normal expected range.

The inter-quartile range rule is an example of such a method. It uses the first and third quartiles, $Q1$ and $Q3$, of the data to identify outliers. First, the inter-quartile range itself is calculated as $IQR = Q3 - Q1$. Then, this value is multiplied by a constant—typically 1.5—and added to the third quartile and subtracted from the first quartile. This produces the interval $[Q1 - 1.5 \cdot IQR, Q3 + 1.5 \cdot IQR]$, which identifies which data points should be preserved. Values falling outside that range are considered outliers and removed.

Feature scaling

Many unsupervised learning algorithms rely on a measure of distance between points, such as Euclidean distance, when training a model. It is therefore important for different dimensions or attributes to be normalised to the same or a similar degree. If one attribute were to have a much larger range of values than all others, it would dictate the outcome of the distance calculations, which in turn would disproportionately influence the model's training.

Two of the most used scaling techniques are Min-Max and Z-Score normalisation. Min-Max normalisation rescales all features between two values, most commonly 0 and 1. Z-score normalisation rescales features such that they are normally distributed, that is, $\mu = 0$ and $\sigma = 1$, where μ denotes the mean and σ the standard deviation [45].

Feature engineering

Feature engineering either selects exiting attributes or creates new attributes which allow machine learning algorithms to operate more efficiently and provide better results [47]. Feature extraction transforms high-dimensional datasets into datasets of lower dimension which conserve some meaningful characteristics of the original data. This transformation, also called dimensionality reduction, is covered in subsection 2.2.5. Feature selection selects a subset of the attributes in the dataset. It is not uncommon for attributes in high-dimensional datasets to be highly correlated, potentially making them redundant. Removing unneeded features can improve training time and the quality of the final model. Identifying and selecting only attributes which are truly relevant to the current task can further improve performance.

Imbalanced data handling

Imbalances in datasets can be challenging as the algorithm may favour the more prevalent values or classes during training, leading to a biased model [47]. There are several methods which can be used to sample the dataset, in such way that any imbalances will be mitigated. These include oversampling and undersampling [45]. There are several oversampling techniques with random oversampling being one of the most common. Dataset tuples belonging to minority classes are replicated several times in the training data which alleviates the imbalance [48]. Random undersampling works similarly, by removing tuples belonging to the majority class.

2.2.5 Dimensionality reduction

Using high-dimensional data in machine learning applications results in more noise and increases the likelihood of there being redundant or unrelated features in the data [49].

Furthermore, the complexity of the training in ML algorithms is often linked the input dimension of the data, as well as the number of tuples in the input dataset [50]. Reducing the dimensionality of the data can therefore provide more efficient training and better-quality results by reducing noise and removing unnecessary features from the dataset. Indeed, simpler models are less impacted by noise and make it easier to identify which features influence the data the most [51].

Feature selection aims to identify which attributes contain the most information and uses those to train the model, whereas feature engineering creates a set of features which are combinations of the original attributes. Principal component analysis (PCA) and its variations are amongst the most commonly used methods, with others including t-distributed stochastic neighbour embedding (t-SNE) and self-organising maps (SOM).

Reducing the dimensionality of the dataset also allows for visual representations of the data. This can aid in analysis and make the KDD process more effective.

Principal component analysis

Principal component analysis is an exploratory data analysis method typically used for dimensionality reduction, feature extraction and data visualisation. PCA aims to lessen information loss when reducing dimensionality by maximising the variance of the data in the projected space. The technique creates new dimensions called principal components (PCs) that are linear combinations of the original features [52].

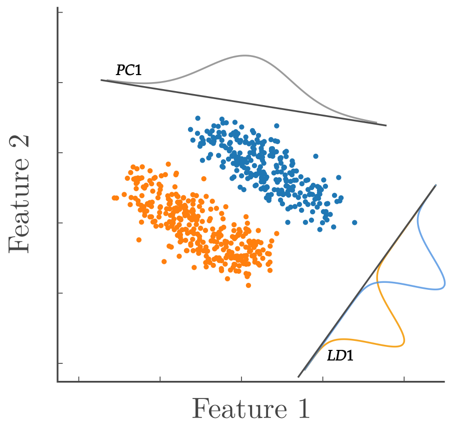


Figure 2.2: PCA vs. LDA. Illustrative example of the differences between PCA and LDA when applied to a two-dimensional dataset. The distributions of data projected onto the first principal component (**PC1**) and first linear discriminant (**LD1**) are shown.

Each principal component is selected to fit the data best and be orthogonal to the previously selected principal components. Principal components are obtained by computing the covariance matrix of the dataset, ensuring each feature has zero mean, and then calculating the eigenvalues and eigenvectors of this matrix. Each normalised eigenvector is then a principal component of the original dataset. To reduce the data dimension from $\mathbb{R}^p$ to $\mathbb{R}^q$ with $q < p$, the first q principal components, sorted in descending order by their respective eigenvalues, must be selected. The data can then be projected onto the selected PCs. The number of principal components used is dependent on both the data and the application. It can be selected by determining what fraction of the total variance is accounted for by each principal component.

Linear discriminant analysis

Linear discriminant analysis is a method typically used for classification and dimensionality reduction tasks. Similarly to PCA, LDA aims to find a linear combination of the input features, explaining the data and reducing its dimensionality. Whilst PCA achieves this by finding axes that maximise the variance of the data, LDA maximises the separability of classes in the dataset. Linear discriminant analysis requires prior knowledge of each datapoint's class, making it a supervised learning technique. The differing objectives of PCA and LDA when applied to the same dataset are illustrated in Figure 2.2

t-distributed stochastic neighbour embedding

t-SNE is a dimensionality reduction algorithm typically used as a way of visualising high-dimensional data. Pairs of points in the high-dimensional space are assigned probabilities, creating a probability distribution where a higher probability indicates similarity between the objects. This step is repeated in the low dimensional space. The Kullback–Leibler divergence which measures how probability distributions differ from each other is then minimised. For datasets with very high dimensionality, this method should be avoided due to the algorithm's time and memory complexities. [53]

Self-organising maps

SOMs are a type of unsupervised learning neural network which produce a low-dimensional representation of a high-dimensional input dataset [54]. SOMs use competitive learning with Euclidean distance as metric to determine which datapoint should influence the network neurons. The representation of neurons it produces, called a map, makes it an ideal tool for visualising high-dimensional data in a low dimensional space and for identifying relationships in the data [55].

Self-organising maps are trained iteratively, as demonstrated in Algorithm 1. At each iteration, an observation is selected at random from the dataset, and a distance metric is used to determine which map neuron is most similar to the selected sample. The winning node coined the best matching unit (BMU) is then updated, along with its neighbouring nodes, to become more similar to the sampled input. The learning rate, which decreases at every iteration, is used as a weight for the update. The neighbourhood function is a further adjustment, with neurons closest to the BMU receiving a more significant update.

Algorithm 1: SOM algorithm

```
Input: $\mathbf{D}$, $\lambda$
initialise weights $\mathbf{W}_v$ for all nodes
$s \leftarrow 0$
while $s < \lambda$ do
    randomly select $\mathbf{D}$(t)
    $u \leftarrow 0$
    $min \leftarrow inf$
    foreach $\mathbf{W}_v \in \mathbf{W}$ do
        $dist \leftarrow d(\mathbf{D}(t), \mathbf{W}_v(s))$
        if $dist < min$ then
            $u \leftarrow v$
            $min \leftarrow dist$
        end
    end
    foreach $\mathbf{W}_v \in \mathbf{W}$ do
        $\mathbf{W}_v(s+1) \leftarrow \mathbf{W}_v(s) + \theta(u, v, s) \cdot \alpha(s) \cdot (\mathbf{D}(t) - \mathbf{W}_v(s))$
    end
    $s \leftarrow s + 1$
end
```

s is the current iteration	$\mathbf{W}_v$ is the weight vector of node v
λ is the iteration limit	u is the best matching unit (BMU)
$\mathbf{D}$ is the input dataset	d is the euclidean distance function
t is the index of a data vector in $\mathbf{D}$	θ is the neighbourhood function
v is the index of a node on the map	α is the learning coefficient

Autoencoders

Autoencoders (AEs) are a type of neural network which aim to learn an encoding of the input data [56]. They do so by attempting to copy their input to their output, having gone through a hidden layer h which has fewer neurons than the input has features. This hidden layer h, often called a bottleneck, forces the model to extract the essential features present in the input data to then be able to reconstruct the input as faithfully as possible. Basic autoencoders are composed of two main elements, an encoder and a decoder, as seen in Figure 2.3. The encoder is used to map the input data to a code or encoding, often called the latent representation. The decoder uses this code to produce a reconstruction of the input. This structure makes autoencoders ideal for dimensionality reduction, as the latent representation will contain the input data's most important features. An autoencoder with two neurons in its bottleneck can then be used to visualise data in two dimensions.

The learning objective for autoencoders is the minimisation of a loss function $L(x, \hat{x})$ which computes the dissimilarity of the input and its reconstruction. An example of a loss function is mean squared error (MSE), as seen in Algorithm 2. The model is then updated using backpropagation which computes the gradient of the loss term with respect to the network's weights and biases. Many autoencoder variants exist with various applications, including de-noising, compression and generative modelling [57].

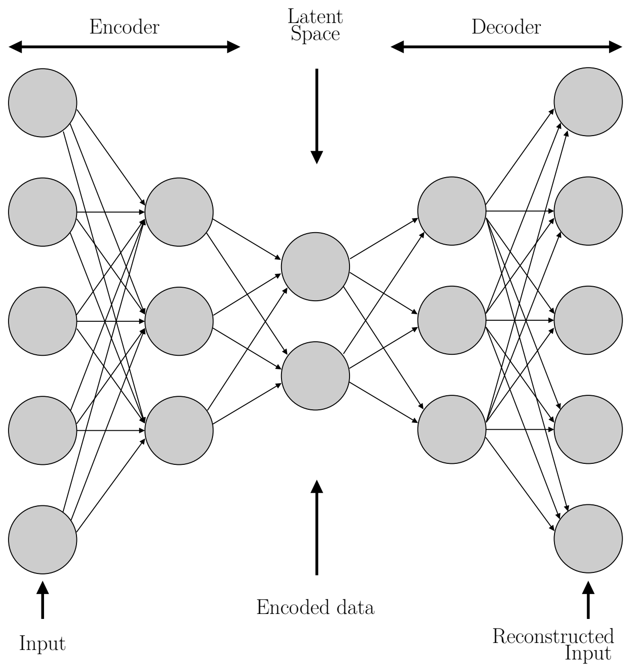


Figure 2.3: Autoencoder neural network. Example of an autoencoder's structure, composed of an encoder, a decoder, and an intermediary latent space which produces the dimensionally reduced data.

Algorithm 2: Autoencoder algorithm

```
Input: $epochs$, $\mathbf{X}$
initialise $\boldsymbol{\theta}_e$
initialise $\boldsymbol{\theta}_d$
while $e < epochs$ do
    $shuffle(\mathbf{X})$
    foreach $\mathbf{x}_i \in \mathbf{X}$ do
        $\mathbf{z}_i \leftarrow enc(\mathbf{x}_i, \boldsymbol{\theta}_e)$
        $\hat{\mathbf{x}}_i \leftarrow dec(\mathbf{z}_i, \boldsymbol{\theta}_d)$
        $loss \leftarrow \frac{1}{n}\sum_{j=1}^{n}(\mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij})^2$
        $\boldsymbol{\theta}_e, \boldsymbol{\theta}_d \leftarrow backpropagation(loss, \boldsymbol{\theta}_e, \boldsymbol{\theta}_d)$
    end
    $e \leftarrow e + 1$
end
```

e is the current epoch	$\boldsymbol{\theta}_d$ are the decoder parameters
epochs is the total number of epochs	n is the input dimension
X is the input dataset	p is the latent dimension
enc is the encoder	$\mathbf{x}_i \in \mathbb{R}^n$ is an input vector
dec is the decoder	$\mathbf{z}_i \in \mathbb{R}^p$ is an encoded vector
$\boldsymbol{\theta}_e$ are the encoder parameters	$\hat{\mathbf{x}}_i \in \mathbb{R}^n$ is the reconstructed input

2.2.6 Clustering

Clustering is an unsupervised learning technique used to group data into a number of categories or clusters [58]. Being an UL task, no labels are available [59] and objects within a cluster must therefore be grouped based on some measure of similarity [60]. Although difficult to define, a cluster can be seen as a group of objects which are similar to one another and dissimilar to objects in other clusters [61].

Cluster analysis is used to determine if a dataset can be summarised by a small number of groups of objects. If the clusters created during training are different enough from one another, they can be used to infer properties of a specific object based on which cluster it belongs to [59].

k-Means

This algorithm uses a measure of distance between points to cluster them. Each point is assigned to the cluster with the closest mean or centroid. Centroids are then updated to be the mean of all points in the cluster. These steps are repeated until the centroids stabilise [62]. While K-means scales well to a large number of samples, it has two well known limitations: the number of clusters needs to be provided and it performs poorly on clusters with irregular shapes. The results obtained by k-means when applied to two different datasets are shown in Figure 2.4.

DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) is an algorithm which identifies clusters as areas where the density of points is high, and different clusters are separated by low density areas [63]. This approach means that it can identify clusters of any shape making it more flexible than k-means and mean shift (Figure 2.4). The algorithm can be applied to large datasets, but either has a high runtime complexity or high memory usage depending on the chosen implementation.

Gaussian mixture models

A mixture model is a probabilistic model which represents subpopulations or groups within an overall population. Gaussian mixture models (GMM) are a sub-class of mixture models which aim to identify several normal distributions, which each model a group within a dataset. The combination of these normal distributions gives the probability distribution of the overall dataset.

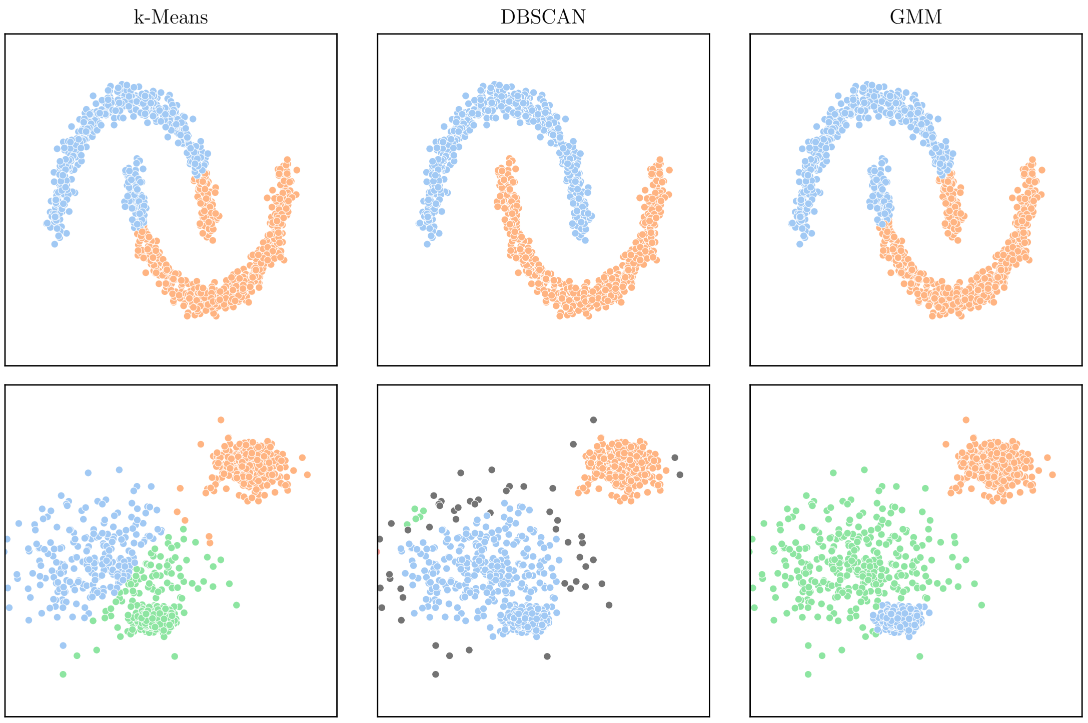


Figure 2.4: Comparison of clustering algorithms. Comparative performance of k-Means, DBSCAN and GMM when applied to two different datasets. Only DBSCAN correctly identifies two clusters distinguishing the crescents (row 1). Three groups of points were used to create the dataset in row 2. These are correctly identified by GMM, whereas DBSCAN identifies only two significant clusters and a number of outliers (grey). k-means detects one cluster correctly (orange) but fails to correctly identify the boundary between the remaining two clusters.

Each point in the dataset can then be assigned to one of these distributions, forming clusters of points (Figure 2.4). Similarly to k-means, this algorithm requires the number of clusters, or, more specifically, the number of mixture components to be specified. The number of components selected can have a considerable impact on the performance and significance of the obtained results.

Chapter 3

System architecture and design

This chapter describes the implemented visualisation and similarity retrieval system's architecture and functionality. Firstly, the chapter presents the essential system requirements (section 3.1). Then, it describes the system design, both in terms of the process which prepares a medical dataset for visualisation and patient similarity retrieval (subsection 3.2.1), and the overall system's architecture, showing the roles and interactions of different components (subsection 3.2.2).

3.1 System requirements

Three principal requirements must be met by a system attempting to use large scale medical datasets. The first is visualisation; that is, the system must present data in an easily interpretable manner. The second is patient similarity retrieval. The system must be able to retrieve existing database entries that are most similar to a new, previously unseen entry. Finally, the third requirement is the ability to interact with the case base—or data currently in the database—for use in case-based reasoning or other applications.

3.1.1 Data visualisation

The availability of electronic health records and large scale medical data has increased over time, with more and more information being recorded. However, whilst more data has become available in electronic formats, it remains challenging to use it. Medical datasets can contain dozens or hundreds of features, making interpreting that data and identifying trends difficult. Figure 3.1 shows what a typical dataset tuple may look like. It includes meta-data components or auxiliary information, features that, for example, may correspond to laboratory test results, and labels that can be used to record symptoms or outcomes. Furthermore, a large number of features—or dimensionality of the data—can make it hard to compare laboratory results or clinical observations between patients. The dimension of the data means that it must be looked at in a tabular format or that only a select few features must be considered and visualised at a

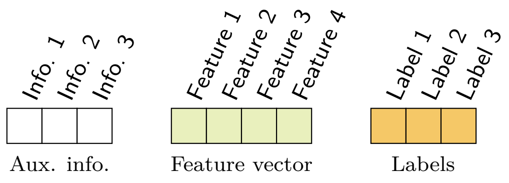


Figure 3.1: Example dataset tuple. It shows the typical structure of each observation in a medical dataset.

time, making it harder to identify similarities or differences between patients as a whole. Therefore, being able to visualise high-dimensional data in a low dimensional space is an essential tool for improving one's understanding of a dataset.

Dimensionality reduction algorithms (subsection 2.2.5) can be used to achieve this goal. A suitable algorithm needs to transform the data to a lower-dimensional space whilst retaining as much meaningful information as possible from the original dataset. Furthermore, seeing as visualisation is a primary objective, one, two or three dimensions are optimal. This allows all data points to be represented onto a single, easily interpretable plot. Two-dimensional visualisations offer a balance of ease of use and understanding, and information retention compared to one or three dimensions. Additional information can be integrated into visualisations by making them animated or interactive. Doing so can make it possible to visualise evolution over time, evaluating trends in the recovery of patients, for example.

Whilst reducing the dimensionality of data does make visualising it possible, it also means that some of the information has been lost. Therefore, retaining information through other means must be considered and would be a requirement for many datasets, otherwise losing a significant portion of their meaning. A potential solution is to accompany visualisations by tabular data containing statistical summaries of the data being represented.

3.1.2 Patient similarity retrieval

Another requirement towards better understanding and utilising medical datasets is the ability to observe one patient and retrieve others who present similar clinical and laboratory characteristics. This retrieval action corresponds to the first stage of the case-based reasoning process (subsection 2.1.2).

Patiently similarity retrieval makes the next steps of case-based reasoning possible, notably the re-use of past information such as diagnostics or treatments plans. Even when these are not available, factors such as medical complications or symptoms shown by a group of patients can be invaluable towards determining if a particular patient is at higher risk or likely to develop comparable symptoms.

3.1.3 Interaction with the case base

A final requirement that builds upon both data visualisation in low dimensional spaces and patient similarity retrieval is the ability to interact with the case base. That is, the creation of an interactive environment that can be used to visualise data points, interact with them, retrieving the most similar ones, and then, where available, display the related outcomes, symptoms or complications.

As well as considering existing cases, the ability to analyse and retrieve similar patients to one who is not currently in the database is a requirement. This means that the dimensionality reduction and retrieval algorithms must support unseen data.

3.2 System design

This section summarises the process used to prepare data for use in visualisation tasks and similarity retrieval (subsection 3.2.1). It then presents an overview of the overall system's architecture (subsection 3.2.2).

3.2.1 Process outline

The process used to reduce data dimensionality for visualisation and to perform similarity retrieval can be summarised into several distinct steps (Figure 3.2): feature selection, preprocessing, dimensionality reduction, and clustering. The obtained representations must then be evaluated to assess the quality of results and ensure they are meaningful.

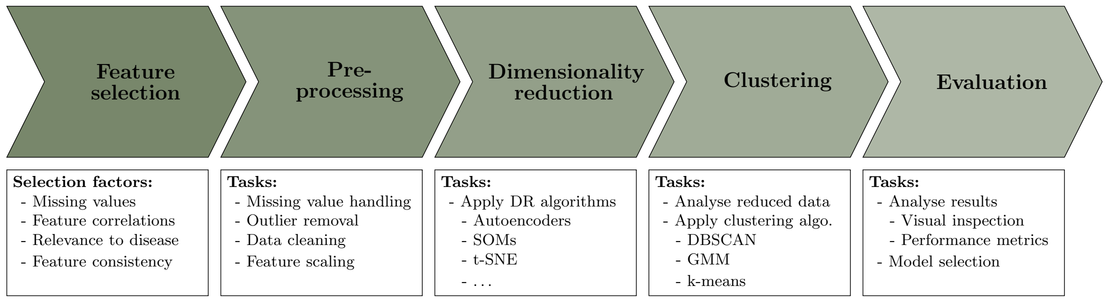

Feature selection
Pre-processing
Dimensionality reduction
Clustering
Evaluation
Selection factors: - Missing values - Feature correlations - Relevance to disease - Feature consistency
Tasks: - Missing value handling - Outlier removal - Data cleaning - Feature scaling
Tasks: - Apply DR algorithms - Autoencoders - SOMs - t-SNE - ...
Tasks: - Analyse reduced data - Apply clustering algo. - DBSCAN - GMM - k-means
Tasks: - Analyse results - Visual inspection - Performance metrics - Model selection

Figure 3.2: Process outline diagram. It summarises the five stages necessary to performing clustering and preparing data for similarity retrieval.

Feature selection

The first step is data selection or determining which features in the dataset are most relevant and likely to produce consistent and relevant results. Features are selected based on their proportion of missing entries, their relevance to the disease being considered or the problem being solved, and how consistent or reliable entries are. Laboratory results, for example, the haematocrit, which is the volume percentage of red blood cells in the blood, do not contain the subjectivity that can arise from visual observations and are, therefore, prioritised to train dimensionality reduction algorithms in this report.

Preprocessing

The next step is preprocessing, where further filtering and preparation is applied to the data. Outliers can be removed using statistical rules, and inconsistencies in formatting can be resolved. The data is scaled to match the input requirements of the dimensionality reduction algorithms, and finally, the data types of some features can be changed where required.

Dimensionality reduction

The processed data can then be used in dimensionality reduction algorithms where the main aim is to represent the data in a format suitable for visualisation and interpretation. Different algorithms, including linear methods such as PCA and non-linear methods such as t-SNE, can be compared using evaluation metrics that indicate if a particular algorithm performs better on a given dataset. Saving the best performing models means that the reduced data they produce can be used with minimal overhead computation in visualisation or potential case-based reasoning applications.

Clustering

The final stage of the process is analysing the arrangement of the data in the reduced space and applying appropriate clustering algorithms. The aim is to determine if medical complications or diagnoses can be inferred based on which cluster a patient belongs to. Where appropriate, clusters can be detected automatically using methods such as k-means or DBSCAN (subsection 2.2.6). Another possibility is to allow the user to define a cluster by retrieving the patients deemed most similar to a selected patient. Doing so provides more flexibility and potentially more accurate results than generic clustering algorithms, which may perform differently depending on the dataset.

Evaluation

The results produced by the dimensionality reduction and clustering algorithms must be evaluated to ensure that the low dimensional representations and clusters produced are meaningful. Indeed, some algorithms such as t-SNE can produce low-dimensional embeddings which seem to contain patterns or clusters even when provided noise as input. Dimensionality reduction algorithms can be evaluated using a series of distance and density metrics (section 5.1). Similarly, the clusters identified by algorithms such as k-means and DBSCAN must be examined to determine if they uncover any sub-groups within the population or information about how points are distributed in the two-dimensional space.

3.2.2 System architecture

The overall system architecture, presented in Figure 3.3, can be divided into three parts that operate in conjunction to fulfil the main system requirements outlined previously. The first is the algorithms and processing used for dimensionality reduction and retrieval of patients, the second is the front end, responsible for data visualisation and the presentation of information, and the third is the back end used for data access.

Algorithms and processing

These algorithms are responsible for reducing the dimensionality of the information in the datasets and retrieving database entries based on similarity measures. The dimensionality reduction algorithms produce models stored in the database where the system back end or other applications can retrieve them. Patient similarity retrieval is then computed in real-time when a request is received from a front end user.

Front end

The user-facing interface, also known as the front end of the system or the client, is the platform with which users can interact to utilise the models described previously. The front end itself is composed of a graphical user interface (GUI) and a JavaScript layer.

The GUI is the interface that users interact with directly and is the point of access for data visualisation and interaction with the case base. For example, it allows users to select an existing patient and visualise the k most similar patients, where the number k is user-defined. Alternatively, users can enter details for a new patient whose details are not in the database and compare said patient to existing database entries. This is equivalent to the “retrieve” stage of case-based reasoning.

The JavaScript layer is responsible for updating the elements visible in the user interface and requesting information from the back end. Visualisations, for example, are generated on the front end using data received from the back end. The JavaScript layer is also responsible for form validation and preparing requests sent to the server using an API.

Back end

The back end, which can also be seen as the data access layer of the system, is responsible for retrieving data from the database and making it accessible to front end users. It comprises a server-side web application programming interface (API) with exposed endpoints that allow users to request resources, a server or processing component that processes these requests, and a database. The requests received by the server are executed, retrieving the required information from the database, applying further processing where necessary and preparing a response to be sent to the front end user. Responses are serialised into a JSON (JavaScript Object Notation) format, which data can be recovered from on the client-side. The database stores both the trained dimensionality reduction models and the complete dataset, containing patient information, laboratory results, and medical observations.

The back-end is also responsible for serving three types of static files. The first is HTML (HyperText Markup Language), which describes the structure of the web page. The second is CSS (Cascading Style Sheets), which specifies design rules, and the third is JavaScript, which is responsible for interactive components and client-side processing.

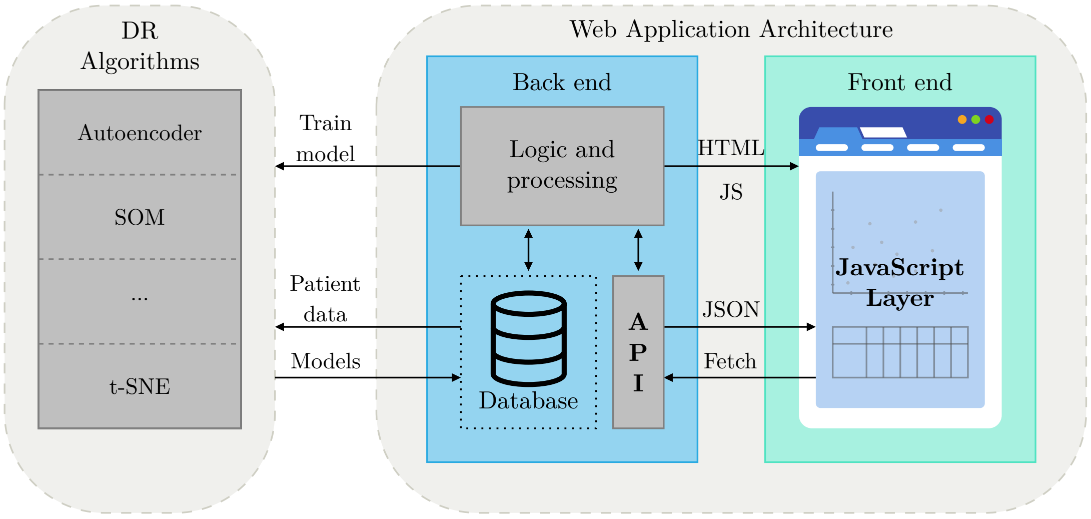


Figure 3.3: System architecture diagram. The implemented system interacts with trained dimensionality reduction algorithms to produce two-dimensional data representations and perform similarity retrieval. These are stored in the database, which is part of the system back end. The back end or server handles requests received from the front end application through an API. Requests are processed on the server, results are then fetched from the database and returned to the front end users through the same API endpoints which received the request.

Chapter 4

Implementation

This chapter presents the key components of the implementation of the patient similarity retrieval and visualisation system. It outlines the implementation process and decisions taken, firstly for the preparation of data (section 4.1), then for the unsupervised learning algorithms (section 4.2), and finally, for the system back end (section 4.3) and front end (section 4.4).

4.1 Data preparation

The data used in machine learning applications needs to be structured in such a way that the algorithms can automatically interpret it. Typically, this means having data in a tidy format where each column corresponds to a different feature and each row is an additional data point. The number of columns is the dimensionality of the dataset, whereas the number of rows gives the number of observations. Subsection 4.1.1 presents the principal dataset used in this report. Subsection 4.1.2 details the cleaning and additional preprocessing applied to the dataset to prepare it for the unsupervised learning algorithms.

4.1.1 Dengue dataset

The primary data source used in this report is an aggregation of datasets collected by the Oxford University Clinical Research Unit (OUCRU) over a series of studies conducted within the Hospital for Tropical Diseases in Ho Chi Minh City, Vietnam [64]. Ten datasets make up the aggregate, with the smallest dataset containing only 75 patients, the largest having over 8000, and the total exceeding 16000 patients. The recorded data spans over twenty years, with the first dataset containing entries recorded in 1999 and the latest entries being reported in 2021.

The aggregated dataset comprises over 400 features, with only seventeen of them having at least one observation in each dataset. The recorded features include administrative information such as the number of days since admission or illness onset, laboratory results such as haematocrit[1] and platelet count[2], patient information such as age and gender, and clinical observations or patient symptoms such as bleeding and shock.

4.1.2 Data preprocessing

The dataset uses a uniform feature naming convention, and feature values have consistent notations due to preliminary cleaning performed by Imperial College London. The dataset also uses a tidy format, making it appropriate for ML applications.

Further data handling, preprocessing and analysis was in large part done using Pandas [65], a data manipulation library for Python, Dataprep [66], an exploratory data analysis library, and several purpose-built utility functions.

4.2 Machine learning models

At the core of the implemented system are the dimensionality reduction mappings or models. They produce two-dimensional representations of the input data, making visualisation possible. Clustering and similarity retrieval is then done in the reduced space where results can be more easily interpreted and analysed.

4.2.1 Algorithms

Autoencoders

Autoencoders were implemented as a parametrised class using the PyTorch machine learning framework for Python [67]. Creating a class facilitates the process of initialising autoencoders with more or fewer layers, different layer sizes, and different latent dimension size.

The encoder takes input numbers in the range between zero and one. The input passes through a user-defined set of fully connected layers, which reduces the number of features to the latent dimension size. The decoder then mirrors the structure of the encoder, using fully connected layers to increase the number of features back to that of the input. A ReLU or sigmoid activation function (Figure 5.15) follows each autoencoder layer. These functions dictate the output of each node and introduce nonlinearity into the network, allowing models to learn more complex functions. A sigmoid activation function follows the final decoder layer, bounding the output into the same [0, 1] range as the encoder input.

The vanilla autoencoder uses mean squared error (MSE) as a loss function, calculating the distance between the input vector and the reconstructed output. The loss terms computed on the training and test sets at each epoch are recorded for further analysis.

[1]Volume percentage of red blood cells in the blood. (%)
[2]Number of platelets within a designated volume of blood. (gigacount/L)

These results can determine if the number of epochs should be increased or decreased, which can impact the model's performance and training time. The ability to create animations showing the encoded data in the latent space at each epoch was also added. Visualising the encoded data can be used to determine if early-stopping of the model's training could be beneficial and provides an insight into how the algorithm trains.

Scikit-learn

The Scikit-learn [68] machine learning library for Python is used on multiple occasions due to its usage of Numpy [69] and Cython [70], which makes it highly efficient and provides a considerable performance improvement over standard Python implementations. The analysis of some dimensionality reduction methods, including PCA and t-SNE, was done using the Scikit-learn implementations. The library was also used for clustering algorithms, including k-means and DBSCAN.

Self-organising maps

The Minisom [71] Python package was used for its Numpy based implementation of self-organising maps. Several utility functions were also implemented to improve the usability of the library, facilitating plotting and result visualisation. These include functions used to plot the differences in values between nodes in the map and the distribution of specific features.

4.2.2 Results analysis

Logs

A logging module was designed to generate logs of the results and parameters used during the training and evaluation of various algorithms and configurations. The generated logs consist of an HTML report, a JSON file used to store data in a machine-readable format, and image files containing any generated plots and figures. The HTML report combines all log elements into a human-readable format where results can be inspected and analysed. Model hyperparameters can easily be saved in a table, along with results produced during training. Saved figures are also inserted into the report. Finally, HTML elements such as text, tables or interactive plots, like, for example, those generated by the Plotly library [72], can be added to the report in a single line of code.

The logger is implemented as a standalone class used as a Python context manager or declared like a traditional Python object. Using a context manager makes it possible for the report to be generated even when the program terminates early with an exception.

Finally, the generated report and associated files can optionally be compressed.

Evaluation techniques

Functions used to evaluate the various clustering and dimensionality reduction algorithms are combined into a single Python module. Using a defined structure makes it possible to use the same functions for all dimensionality reduction methods without adapting them on an individual basis. Each function returns a dictionary that summarises the results in a format that is both human and machine-readable. Where applicable, a figure with plots showing visual representations of the results is also returned. These can then be added to the logs and used to compare algorithms.

For tunable algorithms such as t-SNE, autoencoders and self-organising maps, different parameter configurations were tested using a grid search. A grid search or parameter sweep trains a model with all parameter configurations taken from a finite set of user-defined hyperparameter values. For each configuration, a log file was created, making it possible to determine which parameter configurations performed best.

4.3 Back end

The system backend, which serves data to the front end and interacts with the database, is implemented in Python, an interpreted, dynamically typed language, which makes it ideal for prototyping and rapid development. In addition, its extensive range of libraries and object-oriented programming constructs promote code re-use. The database is responsible for storing the unprocessed data and trained models, whilst an API handles client requests and prepares responses.

4.3.1 Data storage

The database is responsible for storing patient data which has had minimal processing applied. Transformations to the data such as grouping and aggregation are applied at training or query time with other preprocessing steps such as standardisation or outlier removal.

The database also stores trained dimensionality reduction models, which can be accessed when creating visualisation tools and encoding patient data.

4.3.2 Similarity retrieval and visualisation

The back end implementation is centred around an API that manages client requests, allowing users to interact with the case base. The API handles requests for the data used in front end visualisations, the retrieval of similar patients and any further interactions with the database. The server itself is implemented using Flask[3], an application development framework for Python. It handles elements such as routing, receiving requests and sending responses, serialising and deserialising JSON objects, and serving static files.

[3]https://flask.palletsprojects.com/en/2.0.x/

The server-side web-API implements four publicly exposed endpoints accessible via specific uniform resource identifiers (URIs) using HTTP requests (Figure 4.1). Responses use JSON (JavaScript Object Notation) and are sent from the same URIs.

`/get_data`

This endpoint is used to retrieve an array of two-dimensional coordinates, where each point corresponds to one patient's data. The coordinates are obtained by encoding patient information using a dimensionality reduction model. The current implementation uses the model produced by an autoencoder due to the comparative advantages investigated in section 5.2. The client can use the index of each data point in the array to identify a patient when retrieving further information.

`/get_k_nearest`

This endpoint is used for patient similarity retrieval, with each query requiring two arguments. The first being a patient ID, which, as mentioned previously, is the index of the corresponding data point in the data array. The second is a parameter k which indicates that the k most similar patients should be retrieved.

To reduce computation time and, consequently, the APIs response time, brute force kNN (k-Nearest Neighbours) must be avoided. Brute force implementations of kNN use exhaustive search leading to poor time complexity. The algorithm requires iterating over all n points k times, calculating a distance in p dimensions at each step, where p is the dimension of the dataset. This nested loop structure yields a time complexity $\mathcal{O}(knp)$ which is not scalable to large datasets.

K-d trees or k-dimensional trees are multidimensional binary-tree data structures that can efficiently query data for various problems, including kNN [73]. Whilst the data structure does require building time ($\mathcal{O}(pn \log n)$) and additional storage ($\mathcal{O}(pn)$), it reduces querying time considerably for larger datasets. The retrieval of a single nearest neighbour can be done in $\mathcal{O}(\log n)$ time, making the kNN time complexity $\mathcal{O}(k \log n)$.

`/get_k_nearest` uses Sci-kit learns implementation of k-d trees [68]. The tree is built when the server is started and is then stored in memory. For large datasets, the tree could be built once and then saved to the database to reduce computational overhead when starting the server. It could then be updated with insertions and deletions only requiring $\mathcal{O}(\log n)$ time. Retrieval time could be further optimised by taking advantage of the fact that points which are close in the original dataset are close in the tree. This spatial locality reduces the search space, reducing the potential time complexity to $\mathcal{O}(\sqrt{n} + k)$ [74].

The response returned by this endpoint contains the indices of the k points closest to the one specified in the query. An HTML table that contains summary statistics for both the k selected patients and the remainder of the dataset is also included in the response.

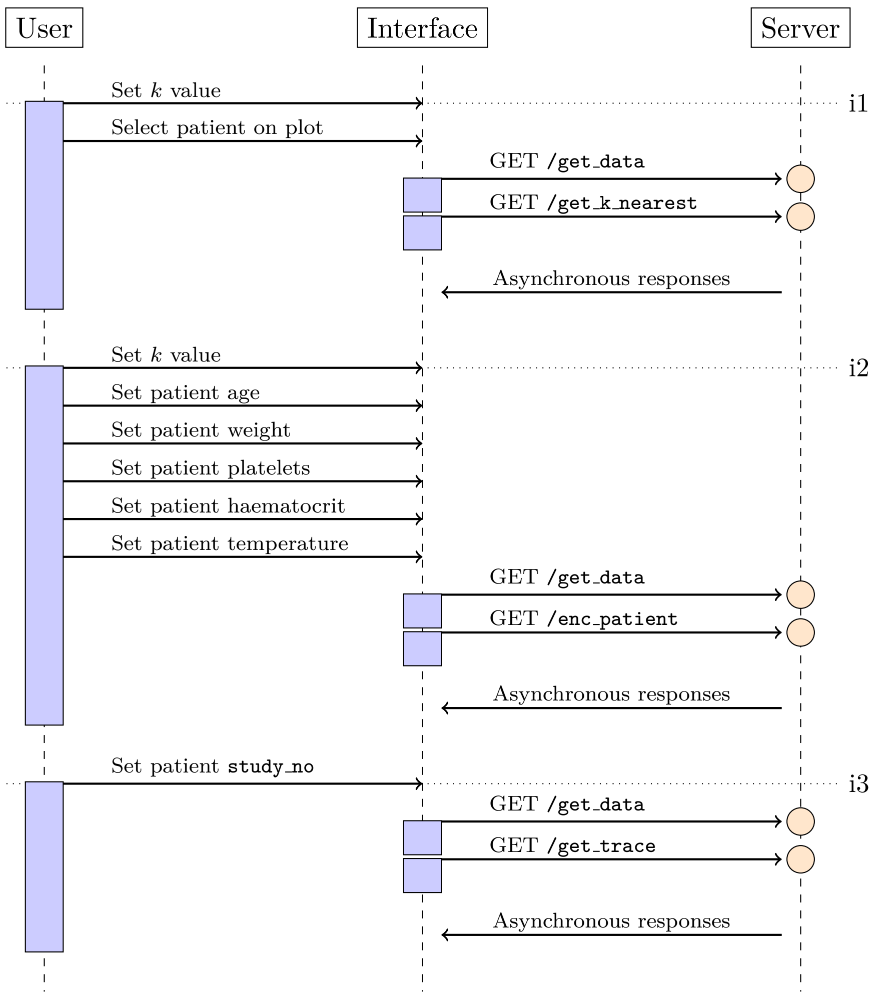


Figure 4.1: Web-API client-server interactions. The web app implements three main interactions with the server which make use of the API endpoints. **i1**, which retrieves the k-nearest neighbours, given a user-selected point on a 2D plot of the dimensionally reduced dataset. **i2**, which retrieves the k-nearest neighbours, given information relating to a new patient using a form. **i3**, which retrieves and displays a patient's 'trace' on a 2D plot. For illustrative purposes, `/get_data` is shown in each interaction; it is, however, only performed once at page load time.

`/enc_patient`

This endpoint is used to visualise the nearest neighbours of a patient who is not currently in the database. Each query to this endpoint must contain the number of neighbours to retrieve k and the patient information required to encode the datapoint, reducing its dimensionality for visualisation and comparison to existing patients. The data required to perform the encoding are the features that were used during training. In the case of the dengue dataset (subection 5.2.1), those features are age, weight, body temperature, platelets and haematocrit. The parameter k and the encoded data point are then used to query the k-d tree and send a response, following the same process as `/get_k_nearest`.

`/get_trace`

This endpoint takes a patient identifier as an optional query parameter. The identifier must be a valid 'study_no' in the dengue dataset used to identify each patient uniquely. If one is not provided, an identifier will be selected at random and returned with the response.

The response sent by this endpoint will contain the dates and coordinates in the two-dimensional space of all observations in the dataset relating to the provided 'study_no'. This is achieved by querying the database for entries matching the identifier. The returned results are encoded using one of the trained models (subsection 4.2.1) and formatted in JSON along with their respective dates.

4.4 Front end

The front end implements an interactive application from where users can visualise and examine patient data. The framework interfaces with the server using the previously described web-API (section 4.3). The visualisation tool consists of two single-page, full-screen apps implemented in HTML (HyperText Markup Language), CSS (Cascading Style Sheets) and JavaScript (JS).

Visualisation

Both applications are centred around an interactive 2D scatter plot implemented using Plotly.js[4], a JavaScript charting library that extends d3.js[5]. Points on the scatter plot represent patients or observations in the dataset and originate from the dimensionality reduction algorithms presented in subsection 4.2.1.

The plots support basic functionality such as panning and zooming, as well as exporting as an image. Further interactive features such as nearest-neighbour retrieval use event handlers to detect click events on the plot and determine which data point is selected.

[4]`https://plotly.com/javascript/`
[5]`https://d3js.org/`

Tabular data

The similarity retrieval application also includes tabular data, which summarises the differences and provides summary statistics about the data included in the current nearest neighbour query and data excluded from the current query. This is done using an HTML table created by the server and returned with each API call to `/enc_patient` or `/get_k_nearest`. The table is then automatically updated with the new data.

Interactivity

As stated previously, interactions with the Plotly.js chart are detected using event handlers[6]. The index of the selected point is obtained, and a query can then be formed. The k value used when retrieving the k most similar patients is input by the user using a slider, whose value updates are detected using its `oninput` event handler. JavaScript's Fetch API[7] is then used to interact with the application's server-side web API. The Fetch API prepares a `Request` object using the query URL and parameters, and then retrieves the `Response` object created by the server. The `Response` object is used to update the data displayed on the plots and other elements such as the HTML table. API calls to the `/enc_patient` endpoint follow a similar process, with the main difference being how the query parameters are obtained. Patient information is entered using a form, and a query is made when the submit button is pressed.

Structure and layout

The front end uses HTML for the page layouts, CSS for styling and JavaScript to incorporate interactivity into the web app. These files are sent to the client when the web page is loaded. Then, further resources and libraries are loaded by the users' browser using content delivery networks (CDNs) or requested from the server using JavaScript. The layout is based on the Octopus theme[8], originally authored by Colorlib[9]. The theme uses the Bootstrap front end framework[10], which contains CSS and JavaScript templates for various layout and interactive elements. The layout followed by each page is shown in Figure 4.2. In addition, Chapter 8 provides a user guide and an overview of the elements on each page.

[6]`https://developer.mozilla.org/en-US/docs/Web/Events/Event_handlers`
[7]`https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API`
[8]`https://github.com/icdcom/octopus`
[9]`https://colorlib.com/`
[10]`https://getbootstrap.com/`

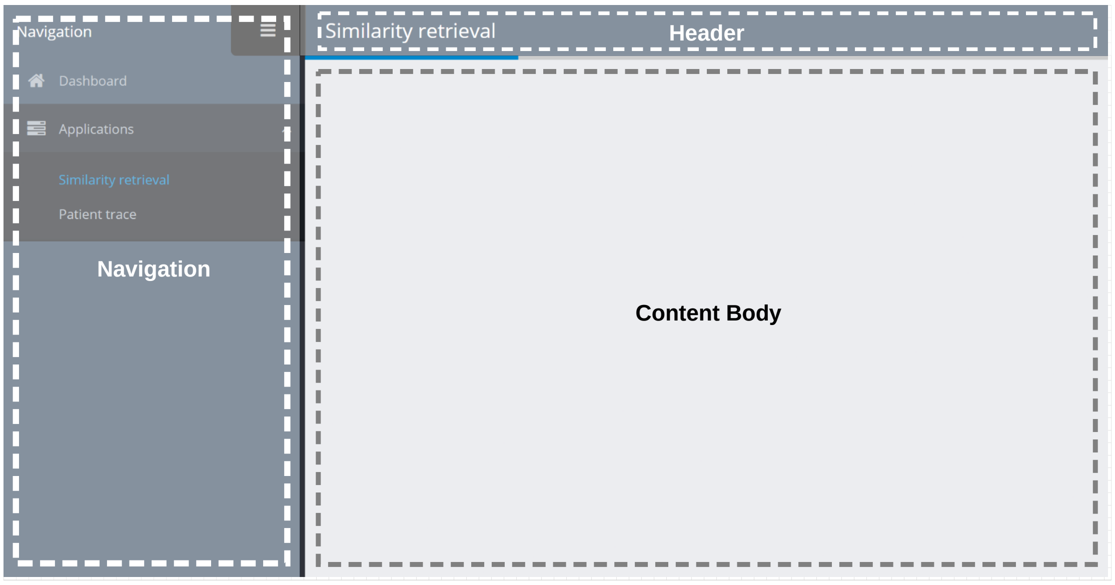


Figure 4.2: Application layout. Each page of the application shares the same layout: a collapsible navigation panel on the left side allows users to access different pages, a header indicates the current page and the content body contains the material to be displayed on the current page.

Chapter 5

Testing and results

This chapter describes the process taken to evaluate the performance of various dimensionality reduction algorithms and the system as a whole. It first presents the metrics used to assess performance and compare algorithms (section 5.1). It then reports the results achieved by various algorithms, aiming to ascertain which is best suited for visualisation and similarity retrieval applications (section 5.2). Experiments were undertaken to evaluate the performance of t-SNE (subsection 5.2.2), self-organising maps (subsection 5.2.3) and autoencoders (subsection 5.2.4). Finally, this chapter outlines the test plan used to evaluate the web app and the performance results obtained (section 5.3).

5.1 Metrics

This section presents the metrics which were used to compare the performance of different algorithms and parameter combinations.

5.1.1 Distance metrics

Dimensionality reduction methods extract the meaningful properties of a dataset and, in the process, lose some of the information. Distance metrics can be used to determine how well the distances between points are preserved. The stronger the relationship between the distances in the reduced space and the distances in the high-dimensional space, the less information has been lost. Computing the distances from each point to every other point can be highly computationally intensive. Therefore, sampling points at random from the dataset to be used in the evaluation is sometimes necessary. The dataset used in the experiments (subsection 5.2.1), for example, has over 14 000 data points. Consequently, using all points in the distance analysis would require 200 million distance calculations in the original high-dimensional space and 200 million in the reduced space.

Sheppard diagram

Sheppard diagrams are scatter plots of two measurements of distances between objects [75]. In dimensionality reduction analysis, the first measurement or collection of distances corresponds to the points in the original dimension. The second measurement is the distances in the reduced space. Plotting one measurement against the other can be used as a visual indication of any distortion incurred when reducing the dimensionality (Figure 5.1). In other words, it shows how well distances have been preserved relative to one another. Collinear points indicate that there has been no distortion. The more points do not lie on this line, the more the distances have been distorted and information lost whilst reducing the dimensionality.

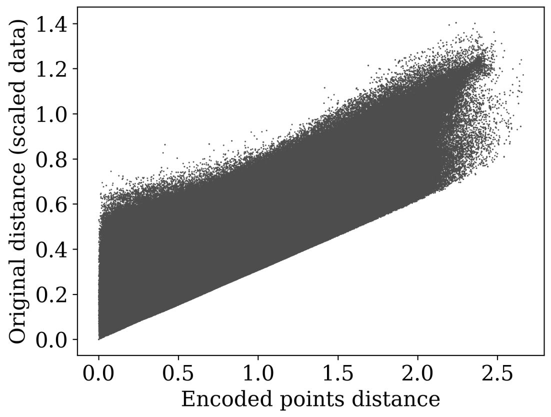


Figure 5.1: Sheppard diagram. It shows an example of a Sheppard diagram. Original distances are plotted against corresponding distances in the reduced space.

Pearson correlation coefficient

The Pearson correlation coefficient is a statistical measure that gives the linear correlation between two variables. The result is given in the [-1, 1] range where 0 indicates no linear correlation [76]. A value of 1 indicates that every increase in one variable is accompanied by a rise of fixed proportion in the other. Conversely, a value of -1 indicates that every increase in one variable is accompanied by a decrease of fixed proportion in the other.

The Sheppard diagram serves as a visual guide that can be used to determine how well the positioning of points have been preserved compared to one another. However, it does not serve as a single metric that can be used to compare the performance of different algorithms. The Pearson correlation coefficient can analyse the linear correlation between the original distances between points and the distances in the reduced space. Values close to 1 indicate that the dimensionality reduction has not caused any significant loss of information and that close together points in the original dataset remain close together in the reduced representation. Values close to 0 indicate that no relationship between the points has been preserved, and a considerable amount of information has been lost during dimensionality reduction.

Spearman rank correlation coefficient

Spearman's rank correlation coefficient measures the dependence between the rankings of two variables [77]. Values of 1, -1 and 0 respectively indicate a monotonically increasing relationship between the variables, a monotonically decreasing relationship and no relationship (Figure 5.2). The measure is defined as being the Pearson correlation coefficient applied to the rankings of the two variables.

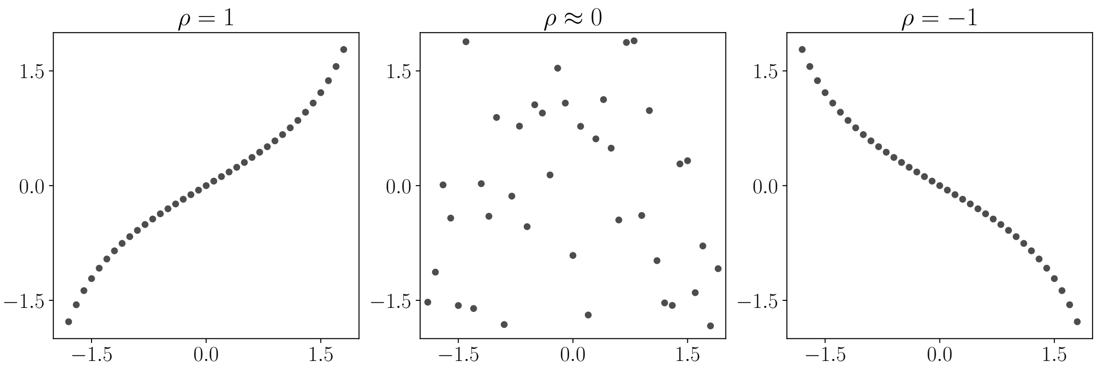


Figure 5.2: Spearman's rank correlation coefficient. Three examples showing how the coefficient value is impacted by the data.

There are applications where maintaining the ordering of distances is more important than having a linear relationship between the distances in the original and reduced spaces. Similarity retrieval, for example, will provide the same results in the original space and the two-dimensional space if the Spearman rank correlation coefficient of the distances in both spaces is one.

Procrustes analysis

Ordinary or classical Procrustes analysis is a statistical method typically used to compare the shapes of two or more objects. The comparison is achieved by performing Procrustes superimposition, which finds a set of translation, rotation and uniform scaling operations which optimally superimposes the objects [78]. An optimal superimposition minimises the Procrustes distance d between objects, typically defined as the square root of the sum of the squared pointwise differences between two objects (equation 5.1). x and y denote two groups of n points in p dimensions. When comparing points with different dimensionality, the dataset with fewer dimensions should have columns of zeros appended to match the dimension p.

$$d = \sqrt{\sum_{i=1}^{n} \left\{ \sum_{j=1}^{p} (x_{ij} - y_{ij})^2 \right\}} \tag{5.1}$$

Procrustes analysis can be used to evaluate the performance of dimensionality reduction algorithms by applying Procrustes superimposition to the original dataset and the data in the reduced space. The final Procrustes distance obtained after the optimal superimposition has been found can be used as a disparity measure between the two sets of points. In this report, the squared disparity is used. A value of zero indicates that the points can be perfectly superimposed and that no information has been lost during the dimensionality reduction process.

5.1.2 Density metrics

Distance metrics alone are not sufficient when comparing different dimensionality reduction algorithms or models. Indeed, as dimensionality increases, the distance from a point to its nearest neighbour nears the distance to the farthest data point [79]. This effect was shown to arise in datasets with as few as ten dimensions [80]. As this happens, all points in the dataset will be at a similar distance from one another, which poses issues when using distances between points to assess performance. Similarity retrieval techniques that rely on distance metrics such as Euclidean distance will also become flawed. The assumption that similar points have similar labels or associated observations no longer holds [81].

Therefore, where available, metrics which make use of labels associated with data points should be used in conjunction with distance metrics when evaluating dimensionality reduction algorithms. Three metrics that evaluate the density of points with a given label compared to all points are used in this report to compare algorithms. Values approaching zero indicate that all points with a given label are close together in the reduced space. In contrast, values close to one indicate that points with a given label cover an area in the reduced space similar to the area covered by all points.

Convex hull ratio

This metric is the ratio of the area of the minimum convex polygon, which encloses all points with a given label, to the area of the minimum convex polygon, which encloses

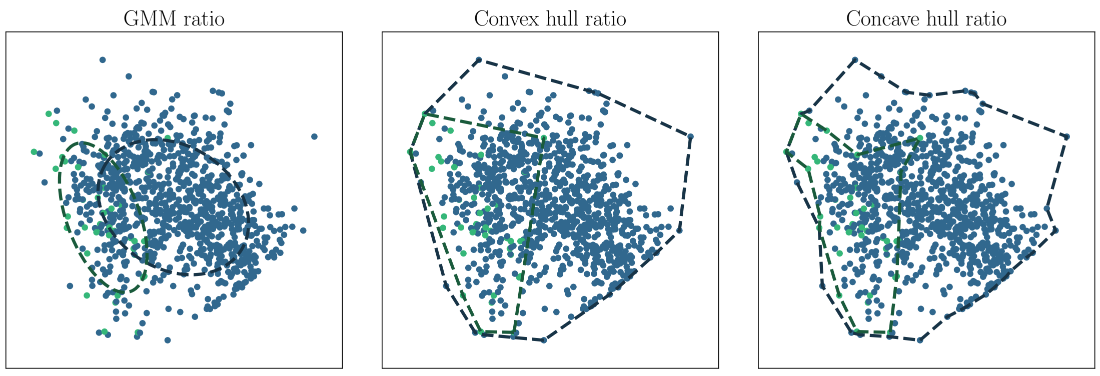


Figure 5.3: Density metrics. The three density metrics used in this report.

all points in the dataset. As shown in Figure 5.3 (convex hull ratio), this ratio is highly impacted by outliers, increasing the convex hull area considerably. Whilst this susceptibility to outliers can make the metric less reliable, it does provide a value that is less likely to change as more points are added to the dataset.

Concave hull ratio

Similar to the convex hull ratio, this metric computes the ratio of the area of a concave polygon, which encloses all points with a given label, to the area of a concave polygon, which encloses all points in the dataset (Figure 5.3). This metric, comparatively, is less impacted by outliers giving a more reliable measure of performance.

GMM ratio

This final ratio uses Gaussian mixture models (subsection 2.2.6) with one component. The metric takes the ratio of the area of the confidence ellipsoid of a model fitted to data points with a given label to the area of the confidence ellipsoid of a model fitted to all data points. This metric is the most robust to outliers as the areas of the ellipsoids are not severely impacted by points that lie far away from the probability distribution's mean (Figure 5.3).

5.2 Experiments

Three experiments were performed to compare the ability of t-SNE, self-organising maps, and autoencoders to produce meaningful visualisations and perform similarity retrieval when applied to a preprocessed version of the dengue dataset.

5.2.1 Dataset

The following experiments all use the Dengue dataset (subsection 4.1.1). The dataset was grouped by patient ID ('study_no') and then aggregated to reduce the number of data points and address the high proportion of missing values.

Each patient's data was forward and backfilled to eliminate any potential issues caused by missing values when aggregating the data. Patient entries were removed in cases where forward and backfilling was impossible due to a patient having no existing values for an attribute. Due to only a limited amount of data being available for patients over the age of eighteen, the data used was limited to those under eighteen. The interquartile range (IQR) rule was applied for features where outliers were an issue, removing entries below $Q1 - 1.5 \cdot IQR$ or above $Q3 + 1.5 \cdot IQR$. The platelets feature, in particular, had values several orders of magnitude above the normal reference range of $150 - 450 \times 10^9/L$ for adults and $150 - 400 \times 10^9/L$ for children, making removing outliers a necessity.

Following outlier removal and missing value handling, the dataset was grouped by patient ID ('study_no') and aggregated, producing one entry per patient. The aggregation functions are outlined in Table 5.1 and were selected to extract the values which are considered most extreme in the context of dengue fever. The haematocrit, for example, is expected to increase when a patient has dengue and become higher as a patient's situation worsens [82]. The maximum haematocrit value is therefore taken for each patient. Similarly, the platelet count is expected to decrease in patients with dengue, and the minimum value is therefore selected. A total of 14484 records remained in the dataset following these preprocessing steps.

A total of thirteen features were extracted from the dataset to be used in the experiments. Their presence in nine of the ten datasets which make up the aggregate makes these features ideal to mitigate the impact of missing values. Five continuous features were selected to be used in training, whilst the remaining eight were selected to analyse and describe the results of the various experiments. Patient age, body temperature, haematocrit, platelets and weight were selected as training attributes. These features are typically recorded on admission and at several points during a patient's stay. The remaining features presented in Table 5.1 were selected as they are amongst some of the most common symptoms and complications of dengue fever.

Table 5.1: Experiment dataset features

Features		**Aggregation**	**Values**
Abdominal pain	boolean	max	4606 (31.8)
Age*	continuous	max	8.0 [5.0,11.0]
Ascites	boolean	max	2331 (16.1)
Bleeding	boolean	max	3724 (25.7)
Bleeding gum	boolean	max	1589 (11.0)
Bleeding mucosal	boolean	max	2666 (18.4)
Bleeding skin	boolean	max	6620 (45.7)
Body temperature	continuous	mean	37.6 [37.2,38.3]
Gender	categorical	first	Female - 6327 (43.7) Male - 8157 (56.3)
Haematocrit	continuous	max	40.3 [37.2,45.0]
Platelets†	continuous	min	169.0 [71.0,243.0]
Shock	boolean	max	701 (4.8)
Weight	continuous	mean	26.0 [19.0,37.0]

Values for continuous features are provided as: median [Q1, Q3]. For Boolean features, the number and percentage of positive observations is given as n (%).
* Patients over the age of 18 were filtered out due to the low number of samples in the dataset.
† IQR rule was applied to remove outliers.

5.2.2 Experiment I: t-SNE

This experiment aims to determine if t-distributed stochastic neighbour embedding (t-SNE) can be used for dimensionality reduction to aid in visualising the dataset and retrieving data points that are deemed similar to one another. The data used in this experiment is the dengue dataset grouped by 'patient' and aggregated to extract the most extreme values for each feature (subsection 5.2.1).

Dimensionality reduction

As shown in the Sheppard diagram (Figure 5.4), distances between points in the original data and points in the two-dimensional data are not well preserved. A Pearson correlation value of 0.769 (Table 5.2) indicates a medium to a high degree of correlation between the original and reduced distances. However, it also indicates that a significant proportion of the distances may not have their ordering preserved and is considerably lower than the Pearson correlation obtained when using PCA (0.925). The diagram shows that points that were very close together in the input space remain close in the reduced space, but that distances are less well preserved for further apart points in the original space. Furthermore, the disparity obtained following a Procrustes analysis on the original points and those in the two-dimensional space is considerably worse than the disparity obtained for PCA.

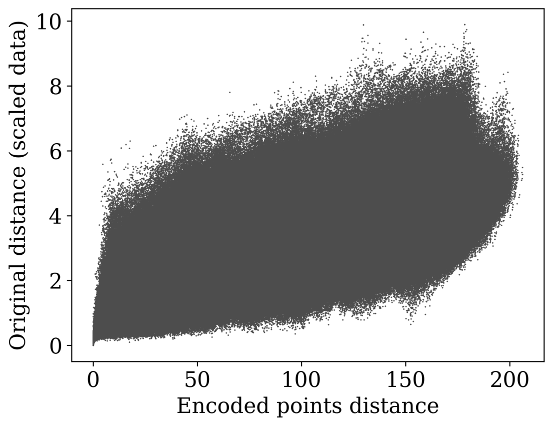


Figure 5.4: t-SNE Sheppard diagram. Distances between points in the original space plotted against the corresponding distances in the low-dimensional space produced by t-SNE

These results are not surprising as t-SNE primarily uses local properties of the data when reducing dimensionality [53]. While t-SNE does retain some of the global structure, making it a helpful tool for further analysing the dataset, it has some limitations when applied to the present use case: identifying and visualising patients similar to a given, previously unseen, patient. Indeed, the algorithm learns a non-parametric mapping, meaning that it does not produce an explicit function which maps the input space to the output, lower-dimensional space [53]. Consequently, traditional t-SNE cannot embed unseen points into the lower-dimensional space. This means that data corresponding to new patients, not used in training, cannot be reduced and visualised in the 2D space. It also prevents patient similarity retrieval from being applied to new data points. Nearest neighbour retrieval can then only be applied to existing points in the embedding, limiting its usefulness which is already impacted by the poor conservation of distances between points. Running t-SNE again with both the new data point and the existing ones can circumvent this issue but has its shortcomings. The t-SNE algorithm has a quadratic

Table 5.2: t-SNE grid search results

Perplexity	**EE**	**LR**	**Pearson**	**GMM ratio**
5	5	100	0.533	0.941[†]
10	5	100	0.750	0.568[†]
20	5	400	0.769	0.456
40	5	400	0.780	0.442
100	40	200	0.777	0.435[†]
200	40	100	0.770	0.416[†]

EE = Early Exaggeration; LR = Learning Rate.
The results presented in this table are those of 6 configurations out of 144 trialled in the grid-search, which are representative of how varying different parameters impacts results.
[†] Configuration only resulted in one cluster.

time complexity $\mathcal{O}(n^2)$, making it inefficient to re-run the algorithm for each new data point. For comparison, parametric algorithms learn a function that can then map an input to a point in a lower-dimensional space, which has a constant time complexity $\mathcal{O}(1)$.

Furthermore, t-SNE has a non-convex objective function which is minimised using gradient descent. Random initialisation means that a local, as opposed to a global minimum, may be found, making results vary between runs, even when using the same data and hyperparameters. Re-training with new data points may further impact results, meaning the produced model needs to be re-evaluated before being used in practice. For t-SNE, which does not aim to preserve distances, visual inspection is usually needed as distance metrics have little meaning, making the process difficult to automate.

Clustering

Whilst t-SNE has its limitations when applied to patient similarity retrieval, it can be an informative tool for analysing the dataset. Finding clusters can be used to identify disease subtypes or determine if a particular group of patients is at higher risk of a specific disease outcome.

The t-SNE algorithm iteratively transforms points creating a representation of the data

Table 5.3: t-SNE grid search hyperparameters

Hyperparameter	**Values**
Perplexity	5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 100, 200
Early exaggeration	5, 12, 20, 40
Learning rate	100, 200, 400

in a two-dimensional space. The results obtained are largely dependant on the hyper-parameters used, which must be carefully selected. A grid search was performed to analyse different configurations, varying the values for three parameters presented in Table 5.3: perplexity, early exaggeration and learning rate. Perplexity adjusts the importance of local versus global features of the data during training. Early exaggeration increases the distance between natural clusters in the original space when the data is reduced to the embedded space. Finally, the learning rate impacts the extent to which points are updated at each algorithm iteration.

Of the 144 parameter configurations used in the grid search, only 15% showed more than one relevant cluster identifiable using DBSCAN (subsection 2.2.6), with most configurations resulting in only one large cluster. The search revealed that higher learning rates performed better when applied to this dataset. Here, performance is measured in terms of the number of clusters found and the density ratios obtained for the shock label. Configurations with a learning rate of 400 resulted in maps with more than one cluster more consistently. Lower learning rates may cause the objective function to get stuck in a local minimum, explaining why higher learning rates consistently performed better. Higher values of early exaggeration (20, 40) resulted in distinct clusters being formed more often than for lower values (5, 12). The lack of distinct clusters for lower values of early exaggeration suggests that clusters in the high-dimensional space—if they exist—are close together. Perplexity values in the interval [25, 50] resulted in more than one cluster for 30% of the parameter configurations. In contrast, perplexity values set below that range resulted in one large cluster due to the global structure of the data being lost. Values above that range resulted in smaller, denser clusters, explained by the increased importance of global topology over local features (Figure 5.5). These values coincide with the original t-SNE paper, suggesting that typical perplexity values lie between 5 and 50 [53]. Furthermore, larger datasets often require a higher perplexity explaining why values over 25 worked best in this case [83].

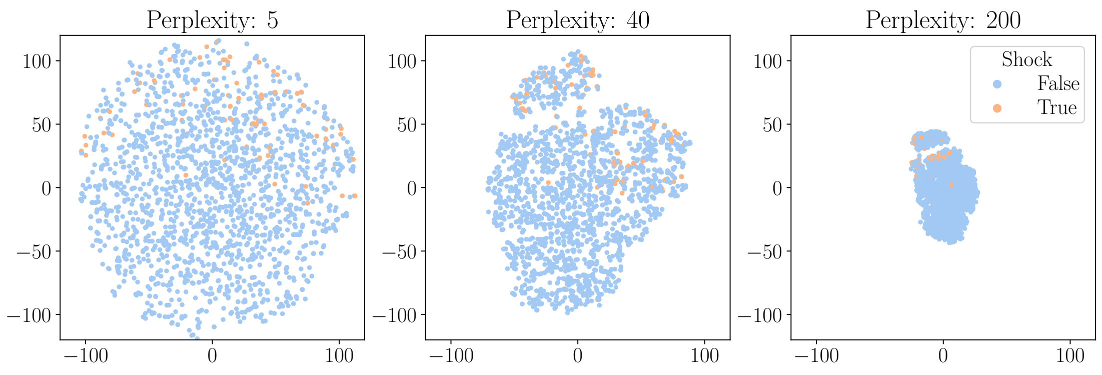


Figure 5.5: Effect of perplexity on t-SNE. This shows how different perplexity values impact the projection of points in the embedded space. Points corresponding to patients with the 'shock' label are also shown.

DBSCAN was used to identify clusters in the embedded space due to its ability to detect clusters of different sizes and shapes. Alternatives such as k-means would be negatively impacted by the proximity of clusters and their varying size. For instance, k-means would identify clusters different from the visually identifiable ones, which is problematic as t-SNE does not necessarily preserve the relative positioning of clusters when reducing dimensionality. Figure 5.6 shows two clusters identified by DBSCAN in the t-SNE reduced embedded space. There is no guarantee that points in Cluster 2 that lie close to Cluster 1 are more similar to points in that cluster than points that lie further away.

While the embedded space produced by t-SNE should not be used for similarity retrieval, the two clusters obtained reveal some information about the dataset. Indeed Figure 5.7 shows that the two clusters identified in Figure 5.6 present considerable differences. Both attributes used in training (Age, Body temperature, Haematocrit, Platelets, Weight) and observations not included in the dimensionality reduction process such as 'Abdominal pain', 'Bleeding' and 'Shock' present significant differences. Indeed, Cluster 1 displays a higher occurrence of all the observations, except for 'bleeding gums'. Cluster 1 also shows a platelet count considerably lower than that of Cluster 2 and a higher haematocrit. This suggests that patients with low platelets and high haematocrit are more likely to show these symptoms.

It must be noted that while Cluster 1 contains a comparatively high proportion of patients with shock (approximately 18%), it only contains about 50% of the dataset's shock patients, with the remainder being in Cluster 2. This is explained by the low

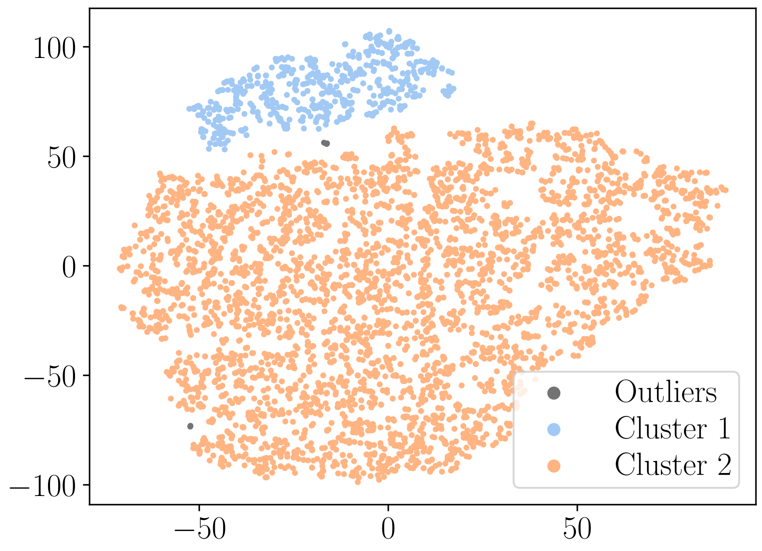


Figure 5.6: t-SNE with DBSCAN clustering. The DBSCAN algorithm detects two clusters and a small number of outliers.

number of observations in Cluster 1 compared to Cluster 2.

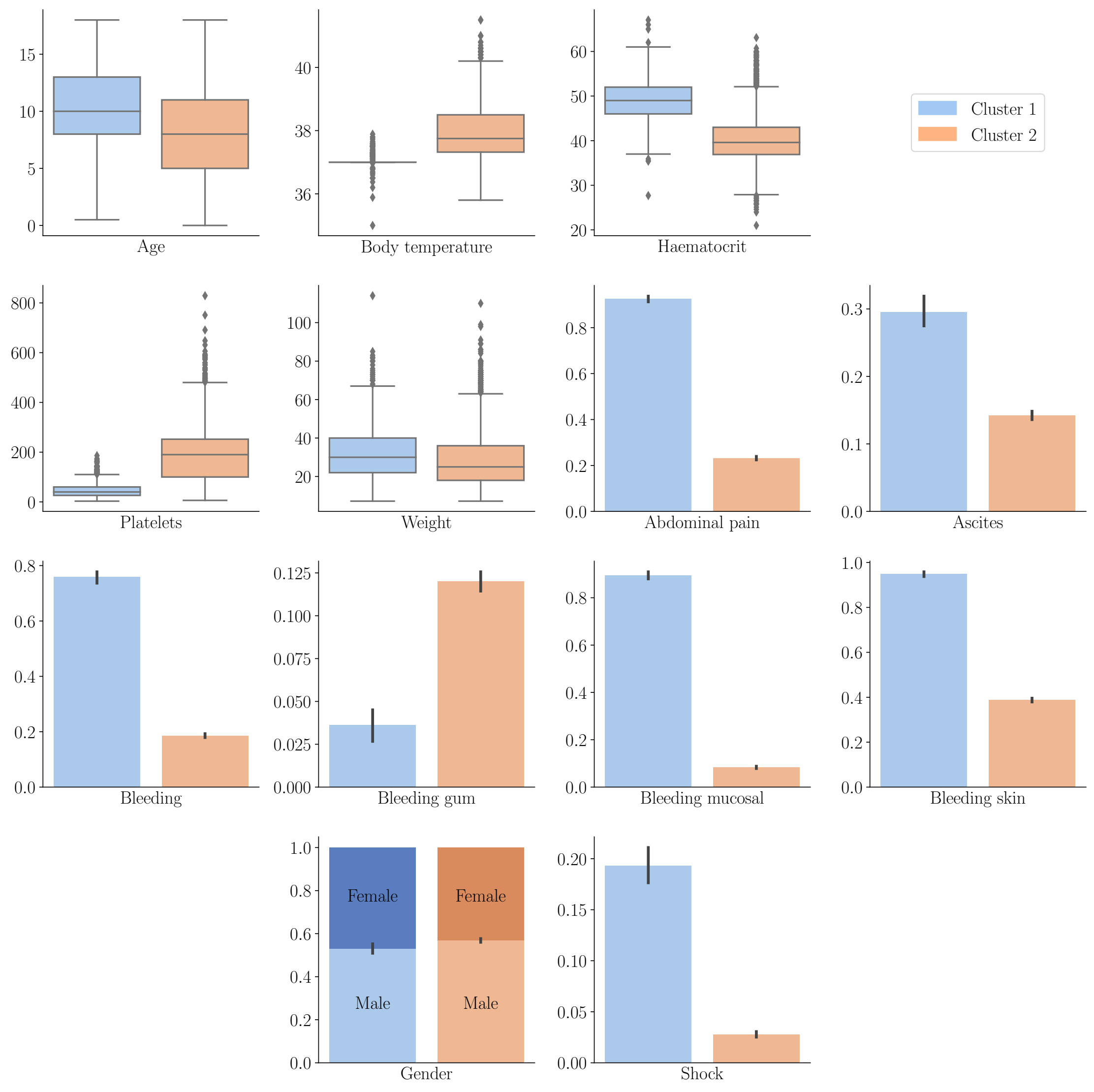


Figure 5.7: Distributions for t-SNE clusters identified with DBSCAN. The distributions of feature values for each cluster identified in the t-SNE embedded space using DBSCAN. The proportion of positive observations is given for binary features such as 'Abdominal pain'.

Insights and results

To summarise, while t-SNE is a powerful dimensionality reduction algorithm, it has limited applicability when the main objective is to perform similarity retrieval on the reduced data. The lack of definitive meaning in the positioning and densities of clusters makes it harder to interpret the data and use distance metrics to identify similar patients. Furthermore, its iterative training learns a non-parametric mapping that cannot map new points to the embedded space. Finally, the quadratic computational time complexity of re-running the algorithm for each new data point further reduces the algorithm's ability to be used in an interactive similarity retrieval and visualisation application. However, the results produced using t-SNE provide some insight into the dataset and can be used for exploratory data analysis.

5.2.3 Experiment II: SOM

This experiment aims to determine if dimensionality reduction using self-organising maps (SOMs) can help visualise the dataset and retrieve patients who are deemed most similar to one another. The dataset used in this experiment is the dengue dataset grouped by patient and aggregated to select the most extreme values for each feature (subsection 5.2.1).

Dimensionality reduction

Self-organising maps have multiple parameters that can influence the quality of results, two of them being the total number of nodes or neurons and the map's dimensions. Jing Tian et al. suggest using $M = 5 \cdot \sqrt{N}$ neurons, where N denotes the number of observations and using the ratio of the data's two largest eigenvalues as the ratio of the map dimensions [84].

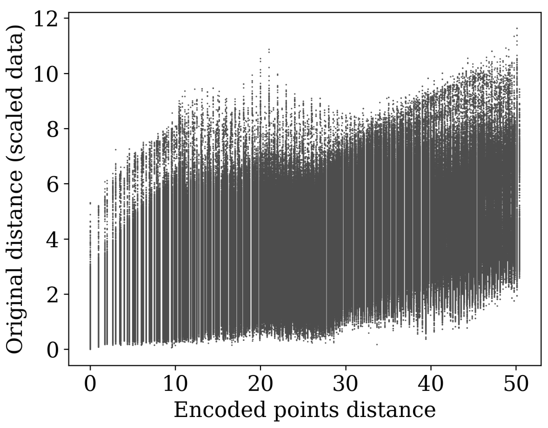
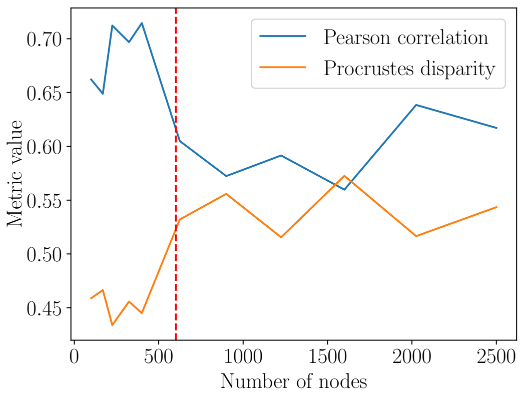


Figure 5.8: SOM Sheppard diagram. Distances between points in the original space plotted against the corresponding distances in the low-dimensional space produced by SOMs.

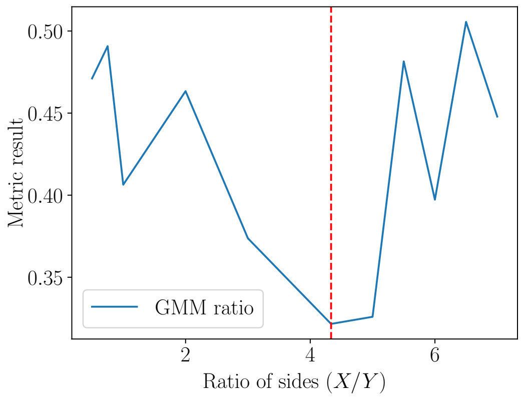
Figure 5.9a shows how varying the number of neurons in the map can impact the preservation of distances between the high and low dimensional spaces. Smaller maps are better at maintaining the relationship between distances in both spaces, as shown by the decreasing Pearson correlation coefficient as the number of nodes increases. Similarly, the Procrustes dissimilarity measure increases with the number of neurons, suggesting that translation, rotation and uniform scaling becomes less effective at minimising the differences between points in the original data and the two-dimensional space. The Sheppard diagram in Figure 5.8 further illustrates how the distances between points and their ordering are affected by dimensionality reduction.

The ratio of the map dimensions was seen to have less impact on the preservation of distances, with most ratios performing similarly for a fixed number of nodes. The poor preservation of distances as the map size increases, can likely be explained by the discrete number of points input observations can be mapped to using self-organising maps. Not only are the distances between points poorly conserved, but the ordering of the distances between points is also affected, as can be seen by the values obtained for Spearman's rank correlation coefficient (Table 5.5).

Varying the number of nodes in the map and the ratio between the length of the sides also impacts how densely packed points with a specific label are. Indeed, Figure 5.9b shows the impact of the ratio of the sides on the density of points with the 'Shock' label. Dimensions that have a ratio that nears that of the ratio of the data's two largest



(a) Impact of the number of nodes in a square map on distance metrics. The number of nodes $M = 5 \cdot \sqrt{N}$ suggested by Jing Tian et al. [84] is shown in red.



(b) Impact of the ratio of the sides of a SOM with a fixed number of nodes on the GMM ratio. The ratio of the dataset's two largest eigenvalues is shown in red.

Figure 5.9: The impact of SOM dimensions on performance metrics.

eigenvalues lead to observations with the shock label being more densely arranged. Moreover, whilst distances are better preserved for maps with fewer than five hundred nodes, points with the shock label appear closer together on slightly larger maps, with maps with six hundred to one thousand neurons performing the best.

A grid search was performed to determine how other hyperparameters impact the two-dimensional maps produced by the SOM algorithm. A total of 224 configurations were evaluated using the parameters in Table 5.4. The three parameters all affect the extent to which nodes are updated at each iteration. The learning rate acts as a weight that decreases as training progresses. The neighbourhood function is used to weight updates based on a point's distance to the BMU (Best Matching Unit). Figure 5.10 shows two-dimensional versions of the neighbourhood functions used. Finally, the sigma parameter determines the reach of the neighbourhood functions, with higher values meaning more nodes will be updated.

The results were analysed using the density of points with the shock label as the principal performance metric. Shock is the most severe observation recorded, and a model which groups patients with shock more tightly can therefore be considered a better one. Analysing the top 20% of best-performing configurations reveals that the Gaussian and

Table 5.4: SOM grid search hyperparameters

Hyperparameter	**Values**
Neighbourhood function	'gaussian', 'mexican_hat', 'bubble', 'triangle'
Sigma	20.0, 15.0, 10.0, 5.0, 2.5, 1.0, 0.05
Learning rate	5.0, 2.5, 1.0, 0.5, 0.25, 0.1, 0.08, 0.05

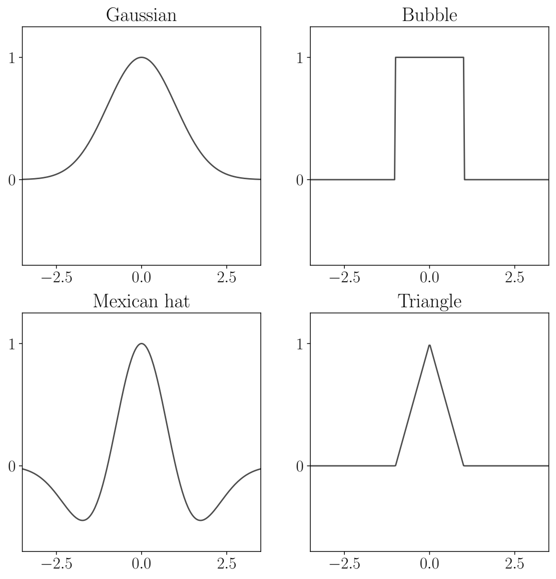


Figure 5.10: SOM neighbourhood functions. Four neighbourhood functions used in the SOM grid search, centred at 0 with sigma 1.

Table 5.5: SOM grid search results

NH function	Sigma	LR	Pearson	Spearman	GMM ratio
Gaussian	1	0.5	0.454	0.446	0.686
Bubble	10	0.25	0.585	0.634	0.355
Gaussian	10	0.25	0.576	0.637	0.320
Bubble	15	0.05	0.633	0.690	0.390
Gaussian	20	0.5	0.748	0.742	0.421

NH = Neighbourhood; LR = Learning Rate.
The results presented in this table are those of 5 configurations out of 224 trialled in the grid search, which are representative of how varying different parameters impacts results.

bubble neighbourhood functions consistently outperform the triangle and Mexican-hat functions. Values of sigma in the 5-15 range led to the best density results. Values lying below that range led to poor distance preservation and points with the shock label being distributed throughout the map. Configurations with a sigma value of 20 resulted in maps with better distance preservation, which can be explained by the higher number of nodes being updated at each iteration, taking more global topology into account. However, their performance in terms of the density of points with the shock label was inferior to configurations in the 5-15 range. Finally, the learning rate was found to have less of an impact, with all tested values lesser than or equal to 0.5 performing similarly well.

Figure 5.11 shows the differences between adjacent nodes in the low dimensional map produced using SOMs and Figure 5.12 shows the distribution of observations with the shock label in the reduced space. The figures show that the reprojections of points are distributed over nearly all nodes and that there are no clear boundaries between areas on the map. This indicates that the algorithm was not able to identify subgroups in the dataset if there are any. While there are no clear separations between areas on the map, Figure 5.12 does show that points with the shock label are primarily grouped on the right side of the map. There are however very few nodes which show a majority of shock patients.

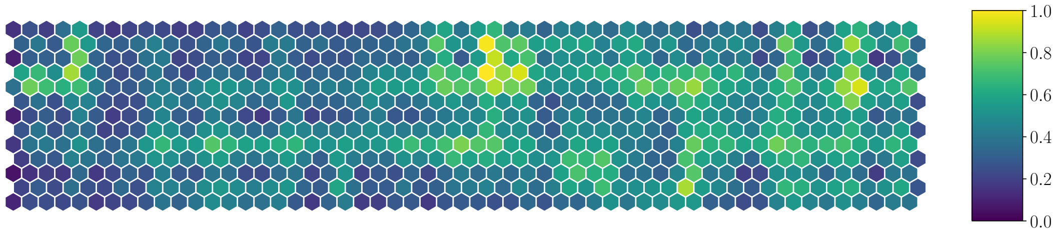


Figure 5.11: SOM node distance graph. It shows how different each node is from it neighbours.

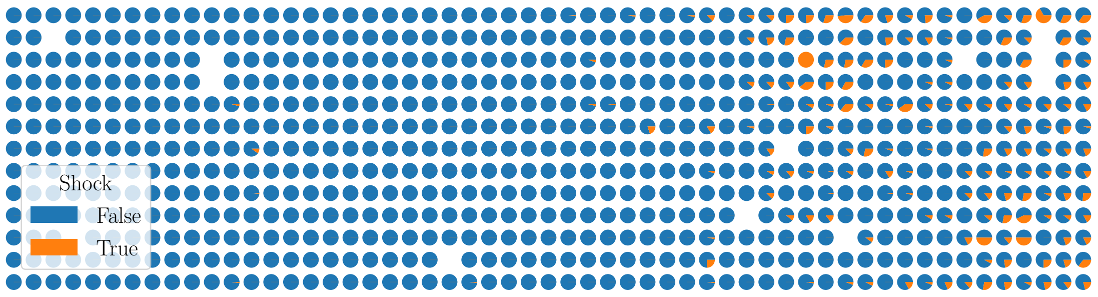


Figure 5.12: Distribution of shock label in map produced using SOM. Pie charts show the proportion of observations which correspond to the shock label for each node in the map.

Clustering

When applied to the dengue dataset (subsection 5.2.1), the two-dimensional mappings produced by the SOM algorithm do not contain any visible clusters. Points are distributed over most map nodes, making techniques such as DBSCAN, which rely on regions of varying point density to identify clusters, ineffective at finding clusters. GMM is similarly affected. Techniques such as k-means are able to identify clusters. However, these are dependant on the shape of the map and not the projection of data points onto the map. Indeed, k-means divides the map into evenly sized sections, as shown in Figure 5.13. While these clusters reveal some information about the dataset (Figure 5.14), they cannot reliably identify disease subgroups or patients at higher risk of developing complications. Therefore, they are of limited applicability to the current application.

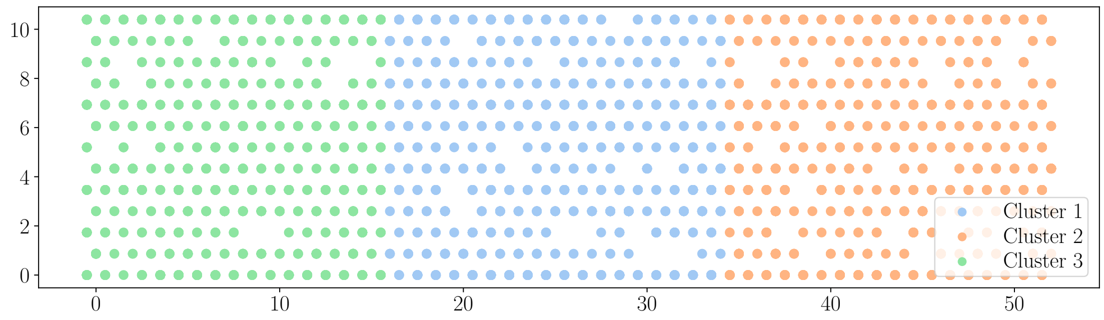


Figure 5.13: SOM with k-means clustering. The k-means algorithm divides the map into three evenly sized clusters as there are no clear clusters or variations in how the data is distributed.

Insights and results

The grid search results illustrate that while distances are generally not well preserved, some configurations yield maps that project points with the same labels close to one another. The poor preservation of distances can largely be explained by the limited number of nodes points are projected onto. While, for the most part, closer points are more similar to one another, the limited number of locations that points can be mapped to restricts the extent to which SOMs can be used for similarity retrieval. Indeed, with many points mapped to the same location, it makes it impossible to distinguish any further between them. Clustering using traditional algorithms is also limited due to the broadly even distribution of points over the map.

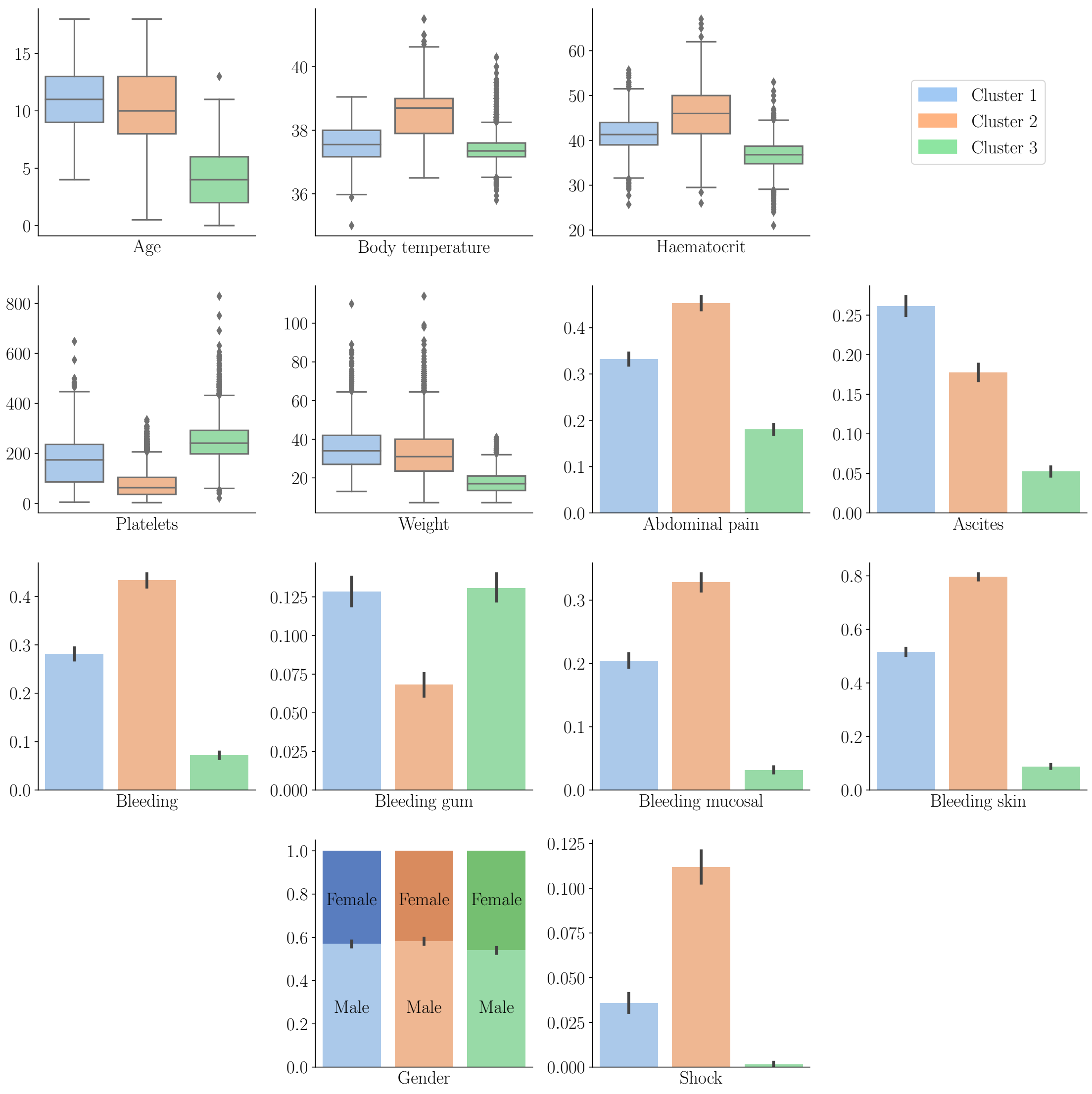


Figure 5.14: Distributions for SOM clusters identified with k-means. The distributions of feature values for each cluster identified in the 2D map using k-means. The proportion of positive observations is given for binary features such as ‘Abdominal pain’.

5.2.4 Experiment III: Autoencoders

This experiment aims to determine if autoencoders can be used for dimensionality reduction to visualise the aggregated dengue dataset (subsection 5.2.1) and perform patient similarity retrieval using the data in the reduced dimension. The experiment focuses on the use of standard autoencoders to ascertain which configurations provide the best performance.

Dimensionality reduction

Autoencoders have multiple parameters which can impact a model's performance, including but not limited to the learning rate, batch size, number of layers and layer sizes, and the number of training epochs. The learning rate dictates the degree to which the neural network's weights will be updated. The batch size determines how many samples are passed through the network before the weights are updated using backpropagation. Bigger batch sizes will mean the model gets updated less frequently but with more significant updates, whereas a small batch size means the model gets updated more frequently but with less significant updates. The number of epochs establishes how many times the complete training dataset will be passed through the network. Finally, the network's hidden layers dictate how complex of a function or mapping the autoencoder can learn. A network that uses only linear activation functions will produce results similar to those obtained using PCA [85]. However, when other activation functions are used, such as ReLU or sigmoid, the model can learn more complex, non-linear mappings. These non-linear mappings can lead to a better reconstruction of the input data, signifying less information loss in the latent space than PCA.

Different parameter combinations were explored using a grid search and compared using several metrics (section 5.1). In total, 1728 different combinations were tested using the hyperparameters shown in Table 5.6. The number of hidden layers in the autoencoder impacts how complex a function it can learn. This directly influences the preservation of distances, with simpler models with fewer layers obtaining distance metric results

Table 5.6: Autoencoder grid search hyperparameters

Hyperparameter	Values
Network layers*	[]†, [5], [4], [3], [5,4], [5,3], [4,3], [5,4,3], [4,4,3,3]
Activation funcitons	'ReLU', 'Sigmoid'
Learning rate	0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001
Epochs	10, 30, 50, 100, 150, 250, 350, 500
Batch size	16, 32

* Network layers refers to the hidden layers used in the encoder. The input layer and latent layer are not included. The decoder layers are the mirror image of the encoder layers.

† Network with no hidden layers other than the latent dimension.

approaching and exceeding PCA's. The model which obtained the highest values for all three distance metrics uses only one hidden layer for the latent dimension followed by a ReLU activation function. The non-linearity provided by the ReLU function allowed the model to obtain a Pearson coefficient value of 0.940, exceeding the value of 0.925 obtained by PCA. Distance preservation is, however, not the only goal. Points with similar labels should be located near to one another, making similarity retrieval applications more meaningful. Models with more hidden layers produced better density metric results, in particular for the shock label. The added complexity introduced to the model by the layers and activation functions allows it to represent high-dimensional data in 2D better but does it at the expense of distance preservation. However, models with too many hidden layers produced representations with datapoints very densely grouped in the latent space, impacting visualisation and utility. The goal of the encoder is to reduce the dimension of the data with each new layer. In this case, where only five input features are reduced to two dimensions, introducing new layers has diminishing returns and can negatively impact the model, as shown in Table 5.7.

Balancing distance preservation and the density metric results produces results covering more of the latent space, improving visualisation whilst preserving utility, with similar points grouped closer together. In this case, this balance is achieved by only using one hidden layer with three neurons in addition to the latent dimension layer, which produces the output. The Sheppard diagrams in Figure 5.16 illustrate differences in the distance preservation achieved by different models.

The choice of activation function also affects the produced model. While distance and density metrics are not heavily impacted by the use of ReLU over sigmoid or visa versa, the two-dimensional representation of the points is affected. Using the ReLU activation (Figure 5.15) can cause neurons to be deactivated, producing straight edges in the latent dimension, which can be hard to interpret (Figure 5.17b). The sigmoid activation (Figure 5.15) avoids this as it does not entirely deactivate neurons by not producing

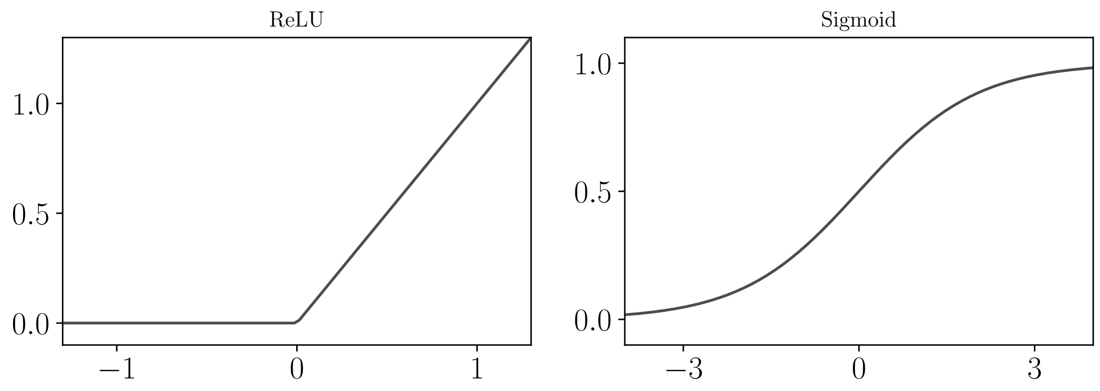
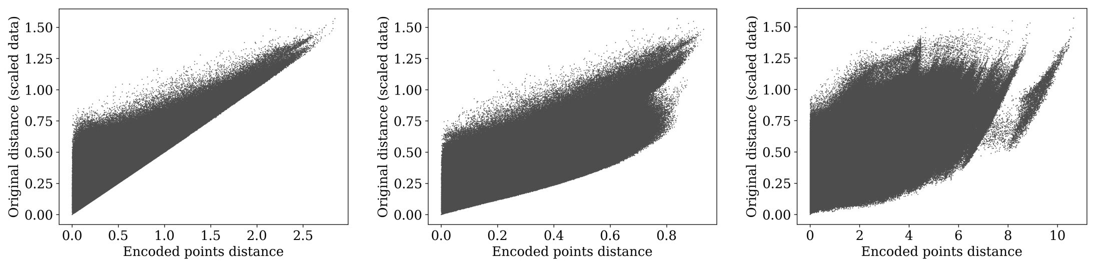


Figure 5.15: Autoencoder activation functions. ReLU ($max(0, x)$) and Sigmoid ($\frac{1}{1+e^{-x}}$) activation functions used in the hyperparameter search.

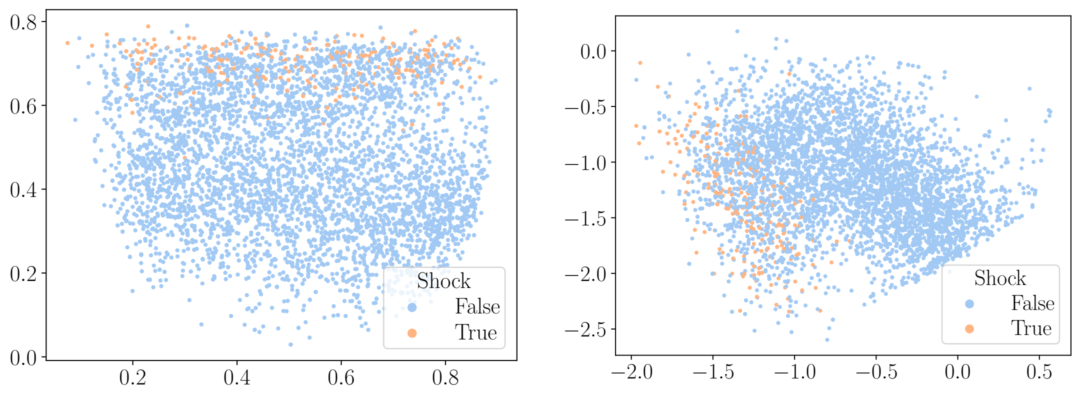
Table 5.7: Autoencoder grid search results

Layers	Activation	Pearson	GMM	Comments
[ ]	ReLU	0.940	0.695	The approximate linearity of this model favours distance preservation.
[ ]	Sigmoid	0.917	0.543	The added non-linearity of Sigmoid affects distance metrics slightly and improves density metrics.
[3]	Sigmoid	0.840	0.321	This model balances distance preservation and density metric results.
[5, 4, 3]	ReLU	0.635	0.104	This complex model has good density metric results but produces dense points in the latent dimension not apt for visualisation.



(a) Layers: [ ] (ReLU) **(b)** Layers: [3] (Sigmoid) **(c)** Layers: [5, 4, 3] (ReLU)

Figure 5.16: Autoencoder Sheppard diagrams. Shows the Sheppard diagrams obtained for autoencoders with different layer configurations.



(a) Sigmoid activations **(b)** ReLU activations

Figure 5.17: Distribution of the shock label in the autoencoder latent space. Examples of the distribution of patients with shock in latent representations produced by different models.

zero values.

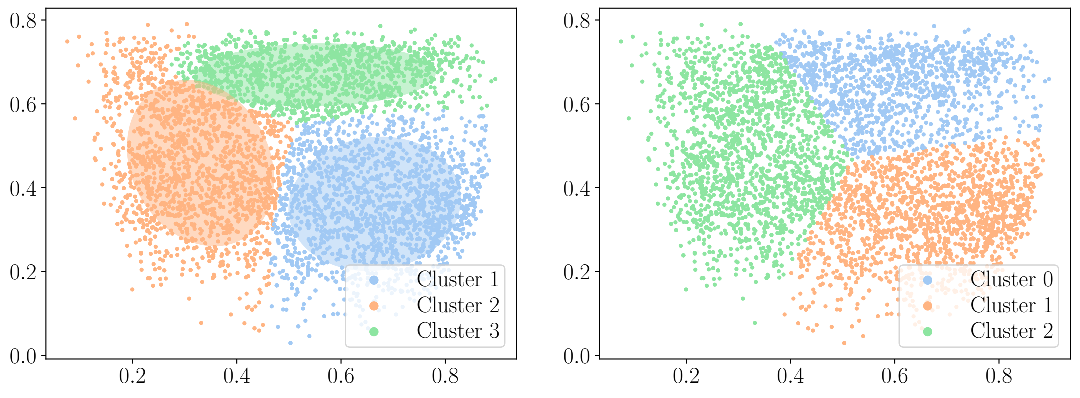
Finally, the learning rate used impacts performance, with higher values such as 0.1 leading to unstable training, preventing the model from converging and producing beneficial results.

Clustering

When using carefully chosen hyperparameters, autoencoders can produce a latent space representation of the dengue dataset (subsection 5.2.1), which preserves the distance relations between points and groups points with the same labels, such as 'shock', closer together. In cases where performance is good when measured by both distance metrics and density metrics, the points in the latent dimension tend to be distributed over a larger area, with no distinctly separated clusters. For example, Figure 5.17a shows points in the latent dimension obtained using one of the best performing hyperparameter configurations. Points are bounded between zero and one in both the first and second dimensions due to a sigmoid activation function being used at the output of the code layer. There are no visibly identifiable clusters, and the points occupy most of the potential area.

The lack of clear separation between groups of points makes using DBSCAN impossible as it relies on regions of high point density and low point density to identify clusters. Whilst points are not uniformly distributed throughout the space, there is no region with a considerably lower point density that separates the points into multiple clusters (Figure 5.17a).

The k-means algorithm can be applied in the latent dimension, but, similarly to self-organising maps, results are not necessarily meaningful or consistent between runs. The



(a) GMM with 3 components **(b)** k-Means with 3 clusters

Figure 5.18: Autoencoder clustering using k-means and GMM. Each colour corresponds to a different cluster identified by the clustering algorithm. The confidence ellipsoids are shown for GMM.

distribution of points in the latent space means that the clusters identified by k-means are primarily dependent on the random initialisation of centroids. The algorithm will divide the points in the latent space into k approximately evenly sized clusters where k is the predetermined number of clusters. That is not to say that k-means cannot identify meaningful clusters. Figure 5.18b, for example, shows clusters identified by k-means. 13.3% of points in Cluster 0 correspond to patients with shock which is considerably higher than the dataset average of 4.8%, indicating that this cluster contains a higher concentration of shock patients than the other two clusters.

As previously mentioned, whilst the data does cover a large portion of the area it is constrained to, it is not uniformly distributed. Thus, it is possible to apply Gaussian mixture modelling (GMM) to identify potential subpopulations or groups of patients within the overall dataset. Doing so with three components leads to the creation of three clusters, as shown in Figure 5.18a. It is immediately apparent that one of the clusters has considerable overlap with the portion of the latent space, which contains the most points with the shock label (Figure 5.17a). Indeed, Figure 5.20 shows that the three identified clusters show considerable differences in the characteristics of the points they contain. Cluster 3, which comprises the highest proportion of patients who suffered from shock (14.9%), has the highest range for haematocrit and the lowest platelet counts. This is consistent with the literature, which lists thrombocytopenia[1] and haemoconcentration[2] as primary factors in diagnosing dengue shock syndrome and dengue haemorrhagic fever [86]. Other factors typically used in diagnosis include bleeding tendencies and plasma leakage, such as the presence of ascites. Cluster 3 shows increased bleeding rates in all categories, except 'bleeding gums', which is considerably lower in this cluster than in the other two. Similarly, the presence of ascites is highest in Cluster 3.

Figure 5.19 demonstrates that the five attributes used in training may not be sufficient to isolate patients with the shock label and group them into a single cluster. The boxplots

[1]Condition characterised by a low platelet count.
[2]Increase in the concentration of red blood cells in the blood.

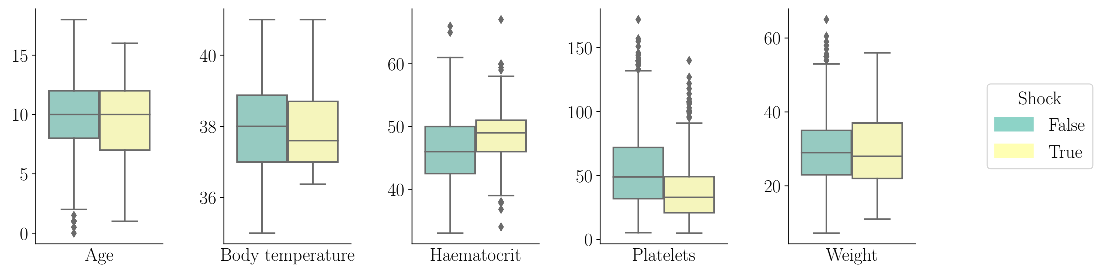


Figure 5.19: Distribution of features within one cluster. Shows the distributions of features for patients with and without shock, within the cluster identified using GMM which has the highest proportion of shock patients (Cluster 3).

of each attribute for patients with and without shock within Cluster 3 show considerable overlap suggesting those two groups have similar distributions. This is consistent with the results found in t-SNE and SOMs, where no parameter configuration and clustering algorithm could form clusters where the prevalence of the shock label exceeded 20%.

Insights and results

To conclude, when adequately configured, autoencoders can produce two-dimensional representations of the dengue dataset, which conserve some of the distance relationships between points and groups similar points close together. The data in the latent space is continuous, unlike self-organising maps, which makes it possible to perform similarity retrieval on any number of points effectively. Furthermore, the parametric model produced during training in the form of weights and biases can be used to encode new, previously unseen data points and represent them in the latent space. The encoding is done in constant time, making a real-time similarity retrieval system possible. Finally, subpopulations in the latent space can be detected in an unsupervised manner using GMM, making it possible to analyse the data further and improve visualisations by making the presence of groups apparent.

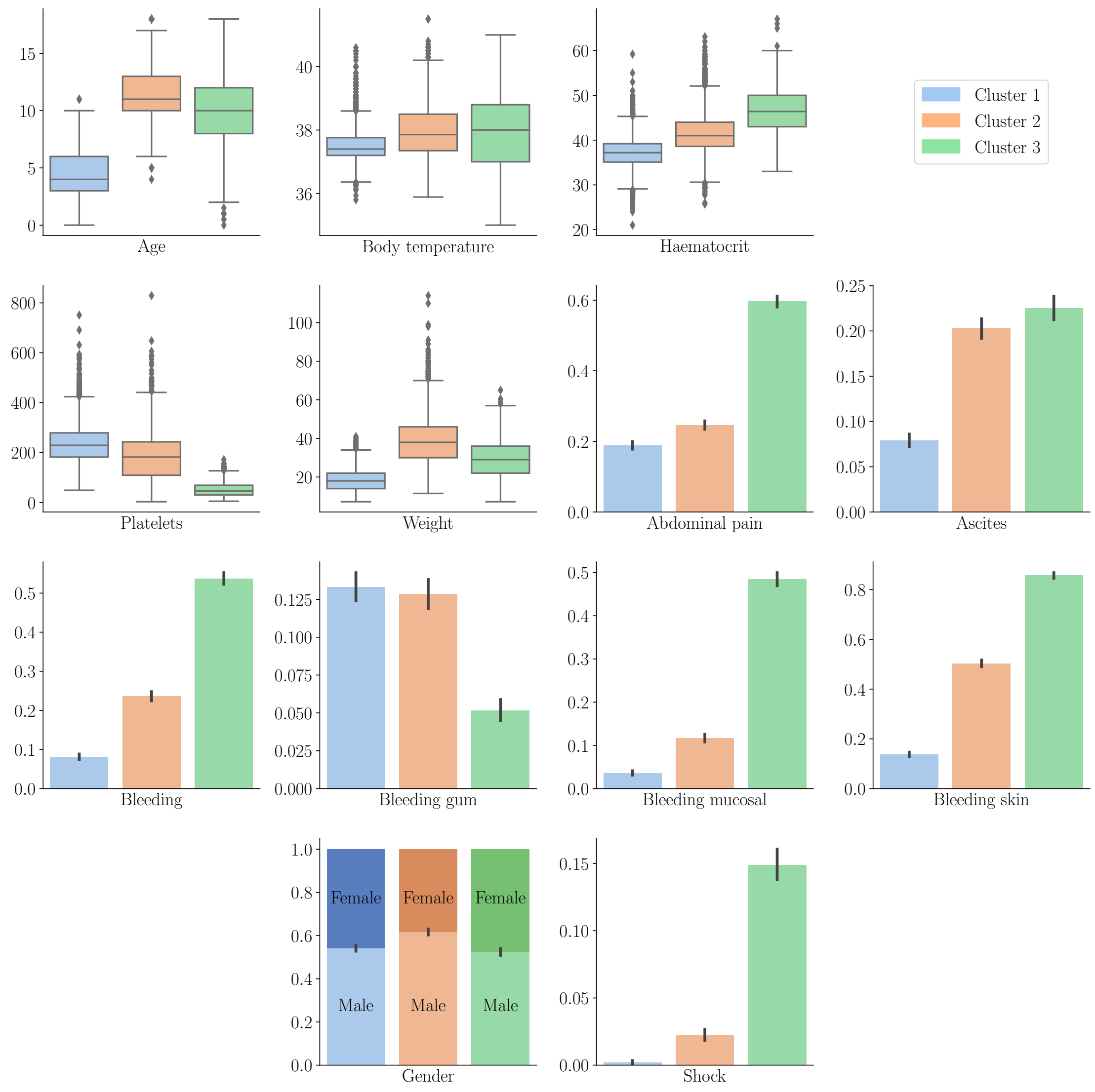


Figure 5.20: Distributions for autoencoder clusters identified with GMM. The distributions of feature values for each cluster identified in the autoencoder latent space using GMM. The proportion of positive observations is given for binary features such as 'Abdominal pain'.

5.2.5 Discussion and findings

The algorithms have been evaluated in terms of their performance using several metrics (section 5.1) and their overall ability to be used in a patient similarity retrieval system. A comparative summary of the results obtained for t-SNE, self-organising maps, autoencoders, and PCA is shown in Table 5.8.

Several requirements need to be met by the two-dimensional space produced by a dimensionality reduction algorithm if it is to be used for patient similarity retrieval. The first is that points that are close to one another should be more similar than points that are far apart from each other. This is measured using the density metrics presented in section 5.1. The clusters identified using different clustering algorithms can also be used to evaluate the similarity of points. Indeed if identified clusters differ considerably in their distributions of features and proportion of observations with a given label, it likely indicates that points within a cluster are more similar to one another than to points outside the cluster. The next requirement is that the system must support real-time usage. That is, there cannot be considerable latency when retrieving similar patients or encoding new patients. Finally, the two-dimensional representation of points must be relatively easy to interpret without considerable background knowledge.

Autoencoders were selected as the algorithm of choice for integration into the patient similarity retrieval system as they can produce models which satisfy all of the above requirements. Indeed, their metric results and performance in clustering indicate that points placed close together in the latent space are similar. Furthermore, the weights learned by the autoencoder during training can then be used to encode new, previously unseen patients into the latent space in constant time. Whilst new patients can be added to the system without retraining the model, this should still be done periodically to ensure the model remains representative of the overall patient population as new entries get added to the dataset. Finally, autoencoders that use a sigmoid activation function at the output of the latent layer produce a representation that can be interpreted without much difficulty. Indeed, autoencoders do not have the complexity of, for example, t-SNE, where the distances between clusters and their size can change considerably with perplexity. Autoencoders also have the advantage of training relatively quickly compared to SOMs and t-SNE.

t-SNE and SOMs were deemed unsuitable for integration into the similarity retrieval system due to not adequately fulfilling all of the aforementioned requirements. Indeed, the non-parametric mapping learned by t-SNE cannot encode unseen points making similarity retrieval using new points impossible to perform in real-time. Furthermore, the low-dimensional representations of data produced by t-SNE, whilst informative, require some background knowledge about the algorithm to avoid drawing invalid conclusions when analysing them. Moreover, the considerable variations obtained when using different hyperparameters or random initialisation mean that several plots need to be considered to make a well-informed analysis of the data.

Self-organising maps present some different limitations, affecting their potential integration into the similarity retrieval system. Indeed, whilst the performance for density metrics rivalled that of AEs, mapping points to a discrete space makes the algorithm less versatile. The limited number of nodes points can be allocated to means that slight differences between points cannot be modelled and that new points must be assigned to one of the existing nodes. Finally, the clustering results obtained for SOMs showed fewer significant variations between clusters than what was identified for AEs and t-SNE, further suggesting that the produced maps may not be suited to similarity retrieval.

This analysis has shown that the performance of dimensionality reduction algorithms is highly dependant on the data being used, and most importantly, the selection of hyperparameters. When applied to different datasets, these algorithms should be re-evaluated. The impact of different dimensionality or distributions of observations may lead to another algorithm or parameter configuration performing better. PCA, for example, is likely to perform better on a lower-dimensional dataset where its linearity is not as negatively impactful.

Table 5.8: Dimensionality reduction and clustering summary

Algorithm	Metrics	Similarity retrieval	Clustering	Final comments
t-SNE	Performance results are highly dependant on hyperparameter selection. The best configuration which identified meaningful clusters obtained a Pearson CC of 0.780 and a GMM ratio of 0.442.	The lack of clear significance in the positioning of clusters and the distance between them impacts the distance-based similarity retrieval. The inability to display new data points using a trained model is also problematic.	t-SNE is the only algorithm to produce distinctly separated clusters and identified a cluster with the highest proportion of shock patients out of all algorithms. This suggests it is a good tool for data analysis.	While t-SNE does create some informative clusters, its inability to reduce the dimensionality of unseen data points makes a real-time similarity retrieval system impossible.
SOM	Performance in terms of distance preservation is poor, in large part due to the discrete space points are mapped to. Density metric results are promising, with a GMM ratio of 0.320 on the best performing map.	The discrete space points are mapped to makes SOM less flexible in selecting the number of similar patients to retrieve. It also makes it impossible to visualise smaller or less significant distances between patients as can be done in a continuous space.	There are no clear clusters in the produced map, and the discrete space points are mapped to makes methods such as GMM and DBSCAN ineffective. k-means does identify clusters with different feature distributions, but these results are dependent on map shape and initialisation.	Whilst density metric results are good, suggesting SOMs are grouping similar points closer together, the limitations imposed by the discrete space limit the applicability to similarity retrieval.
AE	Autoencoders showed good performance for both distance and density metrics, with the selected model achieving a Pearson CC of 0.840 and a GMM ratio for ‘Shock’ of 0.321.	The excellent performance shown for both density and distance metrics makes the latent space produced by the autoencoder appropriate for similarity retrieval. The constant time complexity of encoding new patients is also beneficial.	While there are no visible clusters in the latent space, GMM does identify groups with distinct feature distributions. This suggests that patients within the same cluster are likely to be more similar to each other than to points outside that cluster.	Autoencoders present a balance of good distance preservation and good results for the density metrics. The ability to encode new patients and its support for higher-dimensional data make it ideal for similarity retrieval.
PCA†	Distance preservation is particularly good, with a Pearson CC of 0.925 and a Spearman CC of 0.904. The density metrics, whilst not bad, are inferior to both SOM and AE.	The good preservation of distances makes PCA appropriate for similarity retrieval in low dimensional spaces. However, as dimensionality increases, the distances in the original space become less meaningful, and PCA’s linear nature can negatively impact results.	PCA produces a 2D representation of the data similar to that obtained by AEs. This clustering performance is similar to AEs’ with GMM producing the best clusters.	The performance of PCA is likely to decrease as dimensionality increases due to its linear nature.

CC = Correlation Coefficient.
† Linear algorithm

5.3 Web application evaluation

This section describes the performance and usability metrics used to evaluate the web app and presents the results obtained for different application pages.

5.3.1 Website usability

Several factors can improve the user experience and usability of a webpage or web app. Performance is one of the most influential factors affecting usability, and the metrics used to measure it are presented in section 5.3.2 [87].

Website complexity and ease of navigation are two more factors that have been shown to play a role in usability [87]. Reducing page complexity allows users to find the information they are looking for more quickly, making for a better experience [88]. Better navigability and overall page organisation have a similar effect. Finally, the quality of the content and its usefulness are vital components of a user's experience [89].

The aforementioned factors were all considered during the web app implementation and the design of the graphical user interface (GUI).

5.3.2 Performance metrics

Several metrics can be used to evaluate a website's performance. Obtained results reflect usability and must therefore be taken into account during development and final performance evaluations. The following metrics were selected [90, 91]:

- Page load time: time taken to download and display all website content.
- First content paintful: time taken to display the first piece of content (texts or images) on the webpage. The time taken to display the first element is important as it demonstrates to the user that their query is being fulfilled.
- Time to interactive: time between a user making a request and the page becoming fully interactive. Interactive elements include forms, links, and buttons, amongst others.
- Total blocking time: cumulative time between first content paintful and time to interactive where the page cannot respond to user input.
- Overall weight: total number of bytes transferred to the user. Reducing overall weight can reduce loading times. Doing so through compression, however, can increase the memory usage and processing needed on the client-side.
- Response time: time from a user request to the server response being received. Looking at average and peak response times can indicate if certain types of requests are overloading the server.
- Memory usage: analysing peak memory usage and usage over time can reveal which aspects of the web app need further optimising.

It is important to note that many of these metrics are highly dependent on how and where the web app is hosted. However, they can still provide valuable insight into how different iterations of the web app compare and perform.

5.3.3 Performance results

Table 5.9 summarises the results of the performance metrics for each page of the web app. The app comprises three pages: the dashboard, the similarity retrieval page and the patient trace page. The first is used as a home page for the web app and to navigate to other pages. The similarity retrieval page serves as an interface to perform patient similarity retrieval and visualise results. Finally, the patient trace page is used to visualise the evolution of a given patient during their hospital stay. The metric results were obtained using Lighthouse[3], an open-source website performance analysis tool. Simulated network throttling was used with a throughput of 10Mbps to reflect the average user experience better.

Table 5.9 shows that load times are primarily network bound. Indeed, page load time on an un-throttled connection is twice as fast as first content paintful on a connection with limited throughput. This indicates that a user's experience is impacted, first and foremost, by their network connection. The overall amount of data transferred when loading the site remains low through the use of minified JavaScript and CSS files. Minification removes unnecessary content such as comments and superfluous white space from source files, reducing the file size. Minified source files had their size reduced by 20% to as much as 50%, improving loading times and reducing data usage.

API response times have a considerable impact on user experience. 100ms is considered the limit for a response to user interaction to feel instantaneous, and one second is

[3]`https://developers.google.com/web/tools/lighthouse`

Table 5.9: Web app performance results

Performance metric	**Dashboard**	**Similarity retrieval**	**Patient trace**
Page load time (ms)†	505	1090	1070
First content paintful (ms)	1100	2200	2000
Time to interactive (ms)	1200	2900	2900
Total blocking time (ms)	10	490	490
Response time (ms)‡	-	450-550	15-25
Overall weight (MB)	0.63	2.1	2.1
Maximum memory usage (MB)§	7.3	18.9	19.4

Results are obtained using simulated network throttling with a 10Mbps throughput.
† Page load time is recorded with no network throttling.
‡ Records the typical response time for API calls which can be made from that page.
§ Records the maximum observed memory used by the JavaScript elements on the page.

deemed the upper bound where a user's flow of thought remains uninterrupted by waiting times [92]. Longer response times severely affect user experience, and an indication of progress should be shown to the user to indicate their query is being fulfilled. API calls to the `/get_trace` endpoint are fast enough to be considered instantaneous, whilst API calls from the similarity retrieval page (`/enc_patient` and `/get_k_nearest`) lie close to 500ms, unlikely to affect a users train of thought.

The overall web application evaluation results are promising and suggest that performance is unlikely to be an issue for most potential users. Furthermore, a caching policy is in place, meaning static files and other resources do not need to be retrieved from the server at every visit. This drastically reduces the overall amount of data transferred at each visit, improving load times and, by consequence, user experience.

Chapter 6

Evaluation

This chapter provides a summary and evaluation of the work achieved in this project. It first reports on the dimensionality reduction and clustering algorithms considered throughout this report (section 6.1). It then focuses on an analysis of the system's front end and back end (section 6.2). Finally, the project requirements are revisited, examining how they were addressed throughout this report and in the produced system (section 6.3).

6.1 Algorithm analysis

This report analysed three methods for their ability to be used in dimensionality reduction and visualisation applications: t-distributed stochastic neighbour embedding, self-organising maps and autoencoders. These non-linear algorithms were also briefly compared to PCA, a linear method. This analysis demonstrated that meaningful results can be obtained from all of the above algorithms when adequately configured but that these cannot necessarily be applied to the similarity retrieval task. Indeed, several requirements need to be met to make similarity retrieval worthwhile, as illustrated in subsection 5.2.5.

When evaluating an algorithm's ability to be used for dimensionality reduction, the primary factors were the closeness of points with the same label, the ability to show a previously unseen patient in the two-dimensional space, and the ease of interpretation of the produced space. On this basis, autoencoders were identified as the most suitable algorithm when applied to the dengue dataset. t-SNE was deemed unsuitable due to the necessity to retrain the model for each new patient added. Furthermore, self-organising maps produced promising density metric results but are limited in their ability to be used in similarity retrieval due to the discrete space points are mapped to. However, it was also determined that although not all algorithms were apt for similarity retrieval, applying clustering algorithms to the produced two-dimensional mappings can be informative.

The obtained clusters did consistently show some degree of pertinence, with each algorithm typically identifying one cluster with a heightened proportion of shock patients compared to the dataset average. This cluster typically showed a median platelet count of under $100 \times 10^9/L$ and a heightened haematocrit. In addition, a high proportion of patients with bleeding and ascites were also observed. These observations are all included in the World Health Organisation's clinical case definition for dengue shock syndrome (DSS) [86]. Therefore, even though the identified cluster may only contain 15-20% of confirmed shock patients, the symptoms and laboratory results indicate that the majority of patients in the cluster can be considered similar to one another, even though they may not meet all the criteria defined in the case definition for DSS. This illustrates that all the dimensionality reduction algorithms considered were able to produce meaningful results when paired with an appropriate clustering algorithm for the two-dimensional representation of data produced.

Finally, a noteworthy limitation of the comparative study carried out is that only one dataset was used. The behaviour of different dimensionality reduction and clustering algorithms is likely to be affected when the number of features in a dataset or the distribution of these features is changed. For example, datasets with higher dimensionality are less likely to result in meaningful representations when a linear algorithm is used to reduce dimensionality down to two dimensions.

6.2 Front end and back end analysis

The web-based application makes use of the analysed dimensionality reduction algorithms and the produced similarity retrieval system. It comprises two different data visualisation tools, patient similarity retrieval and patient trace, which provides an overview of a patient's condition over time. These pages facilitate retrieval of relevant data, expediting the decision-making process without considerably altering the standard method used by clinicians. Indeed, diagnoses or treatment plans are often formulated using existing knowledge and past patient data. This system distances itself from a number of current solutions by focusing on data retrieval and not providing any recommendations or automated diagnoses, features which clinicians have criticised for their lack of transparency and potentially unreliable performance.

The web application performs the initially desired task of patient similarity retrieval and visualisation of data. It does so with an average API response time of 500ms for similarity retrieval and 20ms for the patient trace. While 500ms is not low enough to feel instantaneous to users, it is unlikely to disrupt their train of thought or negatively impact user experience. This performance is achieved by using k-d trees, a data structure well suited to nearest neighbour retrieval. The efficient encoding of new patients done by the autoencoder model also has little impact on the final response time.

6.3 Revisiting the requirements

As presented in section 3.1, there are three principal requirements which this project aims to fulfil: data visualisation, patient similarity retrieval and interaction with the case base. The data visualisation aspect is required as medical data, whether in electronic health records or other medical datasets, typically contains many features making it hard to interpret using traditional methods. Reducing the data to a low-dimensional space is therefore essential towards creating a system that can be used to visualise the data retrieved using similarity retrieval. The following requirement is patient similarity retrieval which is also the first stage of case-based reasoning. Retrieving similar patients can help clinicians make decisions using the typical process of analysing different data sources, including past patients' records and outcomes. The final requirement, "Interaction with the case base", is the ability to use and interact with the produced visualisations and patient similarity retrieval system. This requires a user-friendly interface that interacts with the results of the two other requirements. How each of these requirements was addressed is summarised in Table 6.1.

The first two requirements were successfully addressed using dimensionality reduction. Indeed, various algorithms were analysed and compared to determine how meaningful and useful the representations they produce are. Autoencoders were found to group similar patients close to one another in the latent space. Furthermore, the autoencoder learns a function that can encode new patients in real-time, making similarity retrieval possible. Similarity retrieval itself is performed in the latent space using a k-d tree for efficient recovery of nearest neighbours.

The creation of an interactive web application satisfies the third requirement. Indeed, the produced application makes it possible to visualise patients in the two-dimensional space, perform similarity retrieval on either new or existing patients and obtain statistics about the selected group of patients. Furthermore, the 'Patient trace' page makes it possible to visualise how a patient's situation progressed since being admitted to the hospital. A complete overview and user-guide of the implemented GUI is shown in Chapter 8.

Table 6.1: Requirements evaluation summary

Requirement	Method
Data visualisation	Data visualisation in two-dimensions was made possible using dimensionality reduction. An analysis was performed to identify the most suitable algorithm for the dengue dataset. Accompanying these visualisations with the results obtained via clustering enriches the visualisations.
Patient similarity retrieval	Dimensionality reduction algorithms were used in conjunction with kd-trees for fast and efficient retrieval of a customisable number of most similar patients.
Interaction with the case base	An interactive web based application was produced, incorporating elements of both the data visualisation and similarity retrieval requirements.

Chapter 7

Conclusion

This project has developed an interactive patient similarity retrieval system based on a careful analysis of various dimensionality reduction algorithms. The system successfully couples similarity retrieval with visualisations and tabular data to provide an interface that can analyse similar patients, symptoms, laboratory results and demographics. The resulting platform augments the clinical decision-making process by facilitating and accelerating the retrieval of relevant information, making it accessible in an interpretable format.

The analysis of dimensionality reduction algorithms concentrated on extracting the essential features in a dataset, making the data visualisable in a low-dimensional space, and the applicability of the method in a similarity retrieval context. Several metrics and clustering algorithms were used to evaluate the low-dimensional embeddings produced, verifying their ability to be integrated into the completed system. The discovery of clusters with considerable differences and sound feature distributions validates that nearby patients in the two-dimensional space are more similar than distant patients, confirming that the produced results are applicable to patient similarity retrieval.

The produced application was developed with ease of use and navigability in mind. It supports similarity retrieval using both existing and previously unseen patients, in real-time, by applying carefully selected algorithms and data structures such as autoencoders and k-d trees. Moreover, the creation of an extendible web-API improves versatility. It opens the door to integration into a complete clinical decision support system, performing stages of case-based reasoning beyond the retrieval of relevant cases.

7.1 Further work

The patient similarity retrieval problem can be approached from different perspectives or extended to support additional data formats. Indeed, the majority of the methods considered throughout this project all share a similar process. First, the data is preprocessed, with the primary obstacle being the dimensionality of the data. Traditional clustering algorithms are then applied to the cleaned and dimensionality reduced data to produce clusters. These can then be analysed to identify any significant patterns in the data, with the resulting models being integrated into the application.

Temporality

While the 'Patient trace' application page uses temporal information when visualising data, no temporal features are utilised during training. Features such as the time since admission to hospital or the onset of illness could prove helpful when comparing patients. Indeed, knowing if a patient's situation is stable, worsening or improving can influence treatment decisions. Furthermore, a patient whose situation is deteriorating should preferably be compared to previous patients who shared the patient's trajectory. One possible approach to this problem would be to utilise feature engineering, introducing new features into the dataset. For example, the percentage change in a feature over the first three days after admission could be considered.

Image-based data

Electronic health records can sometimes contain results of scans or x-rays. However, these image-based results can prove particularly hard to analyse and compare as traditional methods used to examine large amounts of data usually rely on plots and tables, which are not well suited to images. This issue can be addressed by adapting the process described in this report. Indeed, autoencoders can be applied to images with impressive results. Using convolutional layers, as opposed to fully connected layers used in this report, makes it possible to identify patterns and specific features in images. Convolutional layers also significantly reduce the number of weights in the neural network, making it possible to use more layers and have more input parameters. The models produced by convolutional autoencoders could then be seamlessly integrated into the existing application.

Higher-dimensional data

While the dengue dataset (section 4.1.1) is high dimensional, most features are only present in a few of the datasets that make up the aggregate. Therefore, a significant focus was placed on feature selection, resulting in only five attributes being used in the dimensionality reduction stage. Applying the developed process to a higher dimensional dataset would likely produce interesting results. Indeed, non-linear methods such as autoencoders, self-organising maps and t-SNE would likely perform increasingly well compared to linear algorithms like PCA as the dimensionality increases.

Chapter 8

User guide

This chapter first describes the installation process (section 8.1), which needs to be followed to run the web application's server. The same process is required to train new dimensionality reduction models or use any of the other developed modules or utility functions. Usage of the web application is then detailed for both the server (subsection 8.2.1) and the front end user interface (subsection 8.2.2).

8.1 Installation

The project code repository is hosted on GitHub (Appendix A) and must be downloaded before proceeding with the installation. This can be done manually on GitHub or using `git clone`[1]:

```
$ git clone https://github.com/ostiff/fyp2020-oss1017.git
```

This project uses Python 3.8 as a primary development language. Running the code, be it for the server or the dimensionality reduction models, therefore requires a valid Python installation. This can be done at a system level or through a virtual environment[2] to facilitate the installation of other libraries and requirements.

Several Python libraries must be installed for all components of the application and other modules to function correctly. These can be installed using pip[3], the package installer for Python, from inside the root directory of the project code:

```
$ pip install -r requirements.txt
```

[1]`https://git-scm.com/docs/git-clone`
[2]`https://docs.python.org/3/tutorial/venv.html`
[3]`https://pip.pypa.io/`

8.2 Web application

8.2.1 Server

The web application server can be started from the root directory of the project as follows, making sure that the virtual environment is activated if one is being used:

```
$ python pkgname/application/server.py
```

The server will be started, and the application made accessible locally in a web browser using the following URL: `http://localhost:5000` or `http://127.0.0.1:5000`.

The server does require access to the cleaned OUCRU dengue dataset and will not operate properly without it.

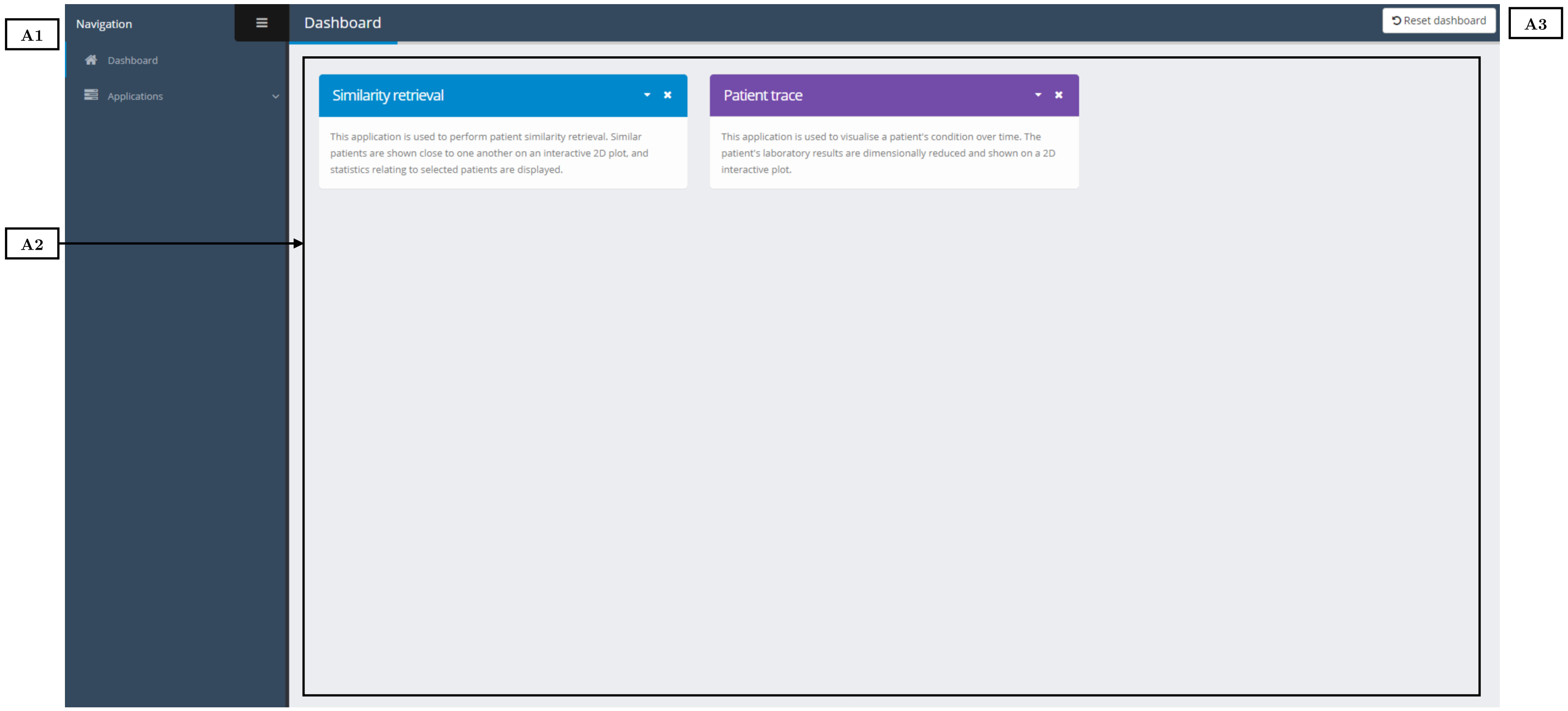
8.2.2 User-interface

The web app comprises three pages: the home page, the similarity retrieval page and the patient trace page. The home page or dashboard serves as a way of navigating to other pages. The similarity retrieval page is used to perform patient similarity retrieval on both new and existing patients. Finally, the patient trace makes it possible to visualise a patient in the latent space over time.

All pages share the same layout, with a navigation panel on the screen's left-hand side, a header indicating the current page and the main content body. An overview of the features and elements of each page is given as follows:

- Table 8.1 shows the dashboard user interface.
- Table 8.2 shows the similarity retrieval page's user interface.
- Table 8.3 shows the similarity retrieval form.
- Table 8.4 shows the patient trace page's user interface.

Table 8.1: Dashboard user guide.



Similarity retrieval

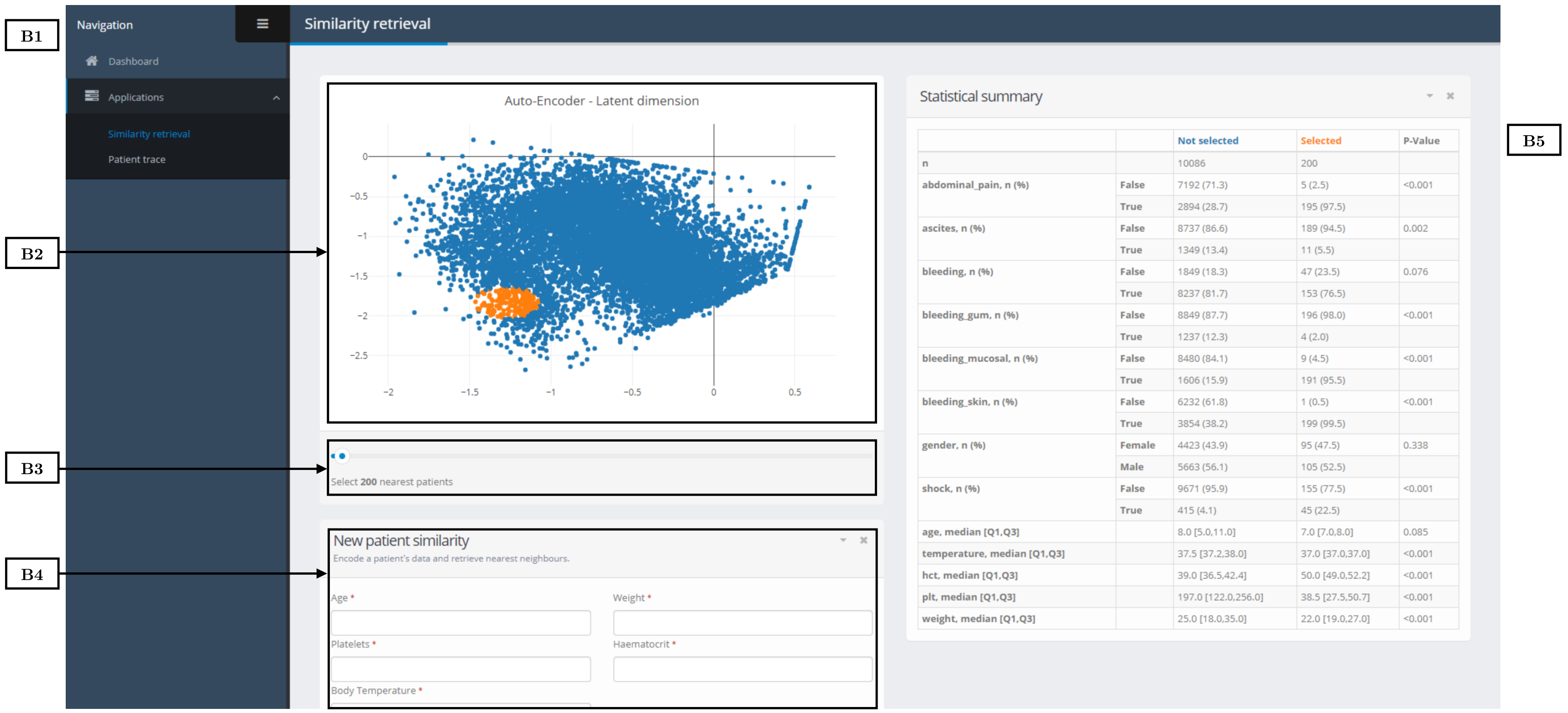
This application is used to perform patient similarity retrieval. Similar patients are shown close to one another on an interactive 2D plot, and statistics relating to selected patients are displayed.

Patient trace

This application is used to visualise a patient's condition over time. The patient's laboratory results are dimensionally reduced and shown on a 2D interactive plot.

[A1] Navigation	[A2] Interactive dashboard	[A3] Reset
Different application pages can be selected using the navigation panel. The panel can be collapsed using the hamburger button in the upper right to increase the space allocated to the main content.	This dashboard shows information about the available pages. The placement of the tiles is customisable and can be changed by dragging and dropping. In the future, more tiles can be added to show details such as alerts or reminders.	This button can be used to reset the dashboard to its original state.

Table 8.2: Similarity retrieval page user guide.

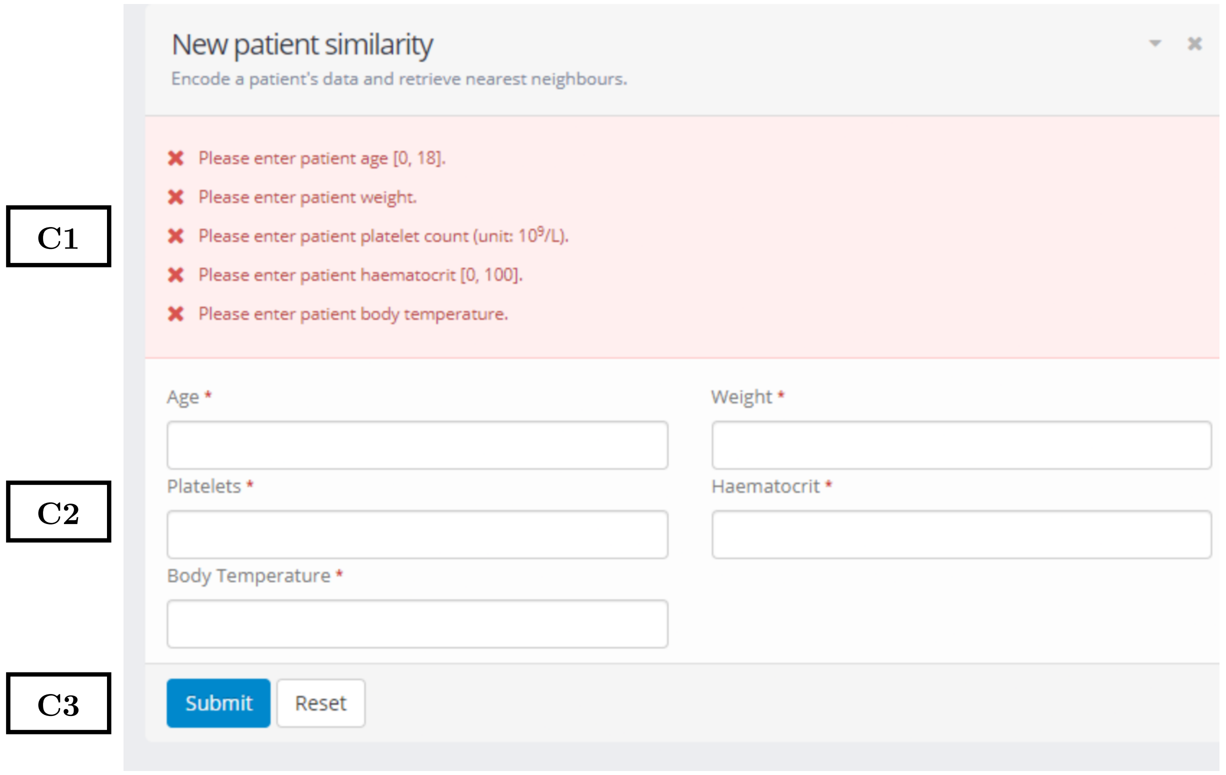


Statistical summary

		Not selected	Selected	P-Value
n		10086	200	
abdominal_pain, n (%)	False	7192 (71.3)	5 (2.5)	<0.001
	True	2894 (28.7)	195 (97.5)	
ascites, n (%)	False	8737 (86.6)	189 (94.5)	0.002
	True	1349 (13.4)	11 (5.5)	
bleeding, n (%)	False	1849 (18.3)	47 (23.5)	0.076
	True	8237 (81.7)	153 (76.5)	
bleeding_gum, n (%)	False	8849 (87.7)	196 (98.0)	<0.001
	True	1237 (12.3)	4 (2.0)	
bleeding_mucosal, n (%)	False	8480 (84.1)	9 (4.5)	<0.001
	True	1606 (15.9)	191 (95.5)	
bleeding_skin, n (%)	False	6232 (61.8)	1 (0.5)	<0.001
	True	3854 (38.2)	199 (99.5)	
gender, n (%)	Female	4423 (43.9)	95 (47.5)	0.338
	Male	5663 (56.1)	105 (52.5)	
shock, n (%)	False	9671 (95.9)	155 (77.5)	<0.001
	True	415 (4.1)	45 (22.5)	
age, median [Q1,Q3]		8.0 [5.0,11.0]	7.0 [7.0,8.0]	0.085
temperature, median [Q1,Q3]		37.5 [37.2,38.0]	37.0 [37.0,37.0]	<0.001
hct, median [Q1,Q3]		39.0 [36.5,42.4]	50.0 [49.0,52.2]	<0.001
plt, median [Q1,Q3]		197.0 [122.0,256.0]	38.5 [27.5,50.7]	<0.001
weight, median [Q1,Q3]		25.0 [18.0,35.0]	22.0 [19.0,27.0]	<0.001

[B1] Navigation	[B2] Latent dimension plot	[B3] *k*-Slider	[B4] Similarity retrieval form	[B5] Statistical summary
Different application pages can be selected using the navigation panel. The panel can be collapsed using the hamburger button in the upper right to increase the space allocated to the main content.	This plot is the result of the dimensionality reduction process using an autoencoder. It shows patients as points in the 2D space were close-together points indicate that patients present similar features. Similarity retrieval can be performed by clicking near a point on the plot.	This slider is used to select the number of patients queried from the database in similarity retrieval. It ranges from a minimum of one patient to the number of patients shown on the plot.	This form is used to perform patient similarity retrieval using data corresponding to a patient currently not in the database. A detailed user guide is provided in Table 8.3.	This table provides a statistical summary of the patients retrieved using similarity retrieval. It compares the selected points to the rest of the dataset for several features, including laboratory results, demographics and symptoms. Statistical significance is shown in the p-value column.

Table 8.3: Similarity retrieval form user guide.

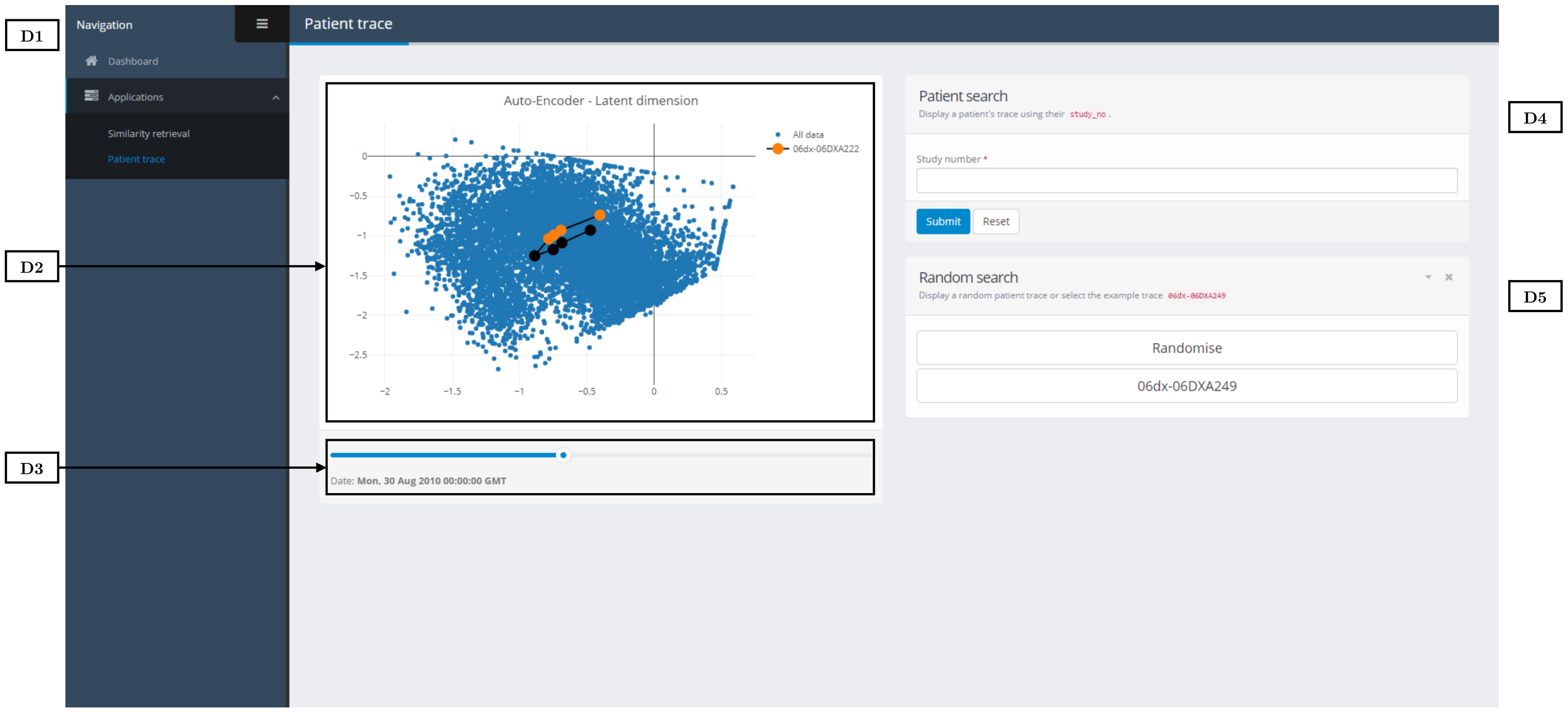


Encode a patient's data and retrieve nearest neighbours.

✖ Please enter patient age [0, 18].
✖ Please enter patient weight.
✖ Please enter patient platelet count (unit: 10^9/L).
✖ Please enter patient haematocrit [0, 100].
✖ Please enter patient body temperature.

[C1] **Validation**	[C2] **Form**	[C3] **Submission**
Form data is verified on submission. If entries are invalid, for example, out of a valid range, the changes to be made are shown here.	Data corresponding to a previously unseen patient can be entered in this form to perform patient similarity retrieval. The number of patients retrieved is set in B3. The features in the form correspond to those used in dimensionality reduction.	The form can be submitted or cleared using these buttons.

Table 8.4: Patient trace page user guide.



Display a patient's trace using their study_no .

Display a random patient trace or select the example trace 06dx-06DXA249

Date: Mon, 30 Aug 2010 00:00:00 GMT

[D1] Navigation	[D3] Latent dimension plot	[D3] Time slider	[D4] Patient search	[D5] Random search
Different application pages can be selected using the navigation panel. The panel can be collapsed using the hamburger button in the upper right to increase the space allocated to the main content.	This plot is the result of the dimensionality reduction process using an autoencoder. It shows patients as points in the 2D space were close-together points indicate that patients present similar features. A patient's 'trace' or evolution over time can be overlayed on the plot.	This slider is used to highlight part of the patient trace. The trace up to and including the selected day will be highlighted.	This form is used to query a patient's data from the database and display it on D2. Patient's can be looked up using their ID, designated 'study_no' in the Dengue dataset.	For illustrative purposes, these buttons can be used to either query a random patient or a predetermined one from the database.

References

[1] J. Liang, Y. Li, Z. Zhang, D. Shen, J. Xu, X. Zheng, T. Wang, B. Tang, J. Lei, and J. Zhang, "Adoption of electronic health records (ehrs) in china during the past 10 years: Consecutive survey data analysis and comparison of sino-american challenges and experiences," *J Med Internet Res*, vol. 23, no. 2, p. e24813, Feb 2021. [Online]. Available: http://www.jmir.org/2021/2/e24813/

[2] C. P. Stone, "A glimpse at ehr implementation around the world: The lessons the us can learn," May 2014.

[3] A. Boonstra and M. Broekhuis, "Barriers to the acceptance of electronic medical records by physicians from systematic review to taxonomy and interventions," *BMC Health Services Research*, vol. 10, no. 1, p. 231, Aug 2010. [Online]. Available: https://doi.org/10.1186/1472-6963-10-231

[4] S. Ajami and R. Arab-Chadegani, "Barriers to implement electronic health records (ehrs)," *Materia socio-medica*, vol. 25, no. 3, pp. 213–215, 2013.

[5] M. R. Cowie, J. I. Blomster, L. H. Curtis, S. Duclaux, I. Ford, F. Fritz, S. Goldman, S. Janmohamed, J. Kreuzer, M. Leenay, A. Michel, S. Ong, J. P. Pell, M. R. Southworth, W. G. Stough, M. Thoenes, F. Zannad, and A. Zalewski, "Electronic health records to facilitate clinical research," *Clinical Research in Cardiology*, vol. 106, no. 1, pp. 1–9, Jan 2017. [Online]. Available: https://doi.org/10.1007/s00392-016-1025-6

[6] I. Sim, P. Gorman, R. A. Greenes, R. B. Haynes, B. Kaplan, H. Lehmann, and P. C. Tang, "Clinical decision support systems for the practice of evidence-based medicine," *Journal of the American Medical Informatics Association : JAMIA*, vol. 8, no. 6, pp. 527–534, 2001.

[7] P. Bountris, M. Haritou, A. Pouliakis, N. Margari, M. Kyrgiou, A. Spathis, A. Pappas, I. Panayiotides, E. A. Paraskevaidis, P. Karakitsos, and D.-D. Koutsouris, "An intelligent clinical decision support system for patient-specific predictions to improve cervical intraepithelial neoplasia detection," *BioMed Research International*, vol. 2014, p. 341483, Apr 2014. [Online]. Available: https://doi.org/10.1155/2014/341483

[8] D. L. Hunt, R. B. Haynes, S. E. Hanna, and K. Smith, "Effects of computer-based clinical decision support systems on physician performance and patient outcomes: A systematic review," *JAMA : the journal of the American Medical Association*, vol. 280, no. 15, pp. 1339–1346, 1998.

[9] B. Hernandez, P. Herrero, T. M. Rawson, L. S. P. Moore, B. Evans, C. Toumazou, A. H. Holmes, and P. Georgiou, "Supervised learning for infection risk inference using pathology data," *BMC Medical Informatics and Decision Making*, vol. 17, no. 1, p. 168, Dec 2017. [Online]. Available: https://doi.org/10.1186/s12911-017-0550-1

[10] R. Nieuwlaat, S. J. Connolly, J. A. Mackay, L. Weise-Kelly, T. Navarro, N. L. Wilczynski, and R. Brian Haynes, "Computerized clinical decision support systems for therapeutic drug monitoring and dosing: A decision-maker-researcher partnership systematic review," *Implementation science : IS*, vol. 6, no. 1, pp. 90–90, 2011.

[11] M. Damhof, "Ps-010 evaluation of a clinical decision support system to optimise cytotoxic drug dosing and continuous surveillance in outpatient cancer patients with renal impairment," vol. 24, 03 2017, pp. A231.2–A231.

[12] I. Carvalho, V. Lima, J. S. Yazlle Rocha, and D. Alves, "A tool to support the clinical decision based on risk of death in hospital admissions," *Procedia Computer Science*, vol. 164, pp. 573–580, 2019, cENTERIS 2019 - International Conference on ENTERprise Information Systems / ProjMAN 2019 - International Conference on Project MANagement / HCist 2019 - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919322690

[13] G. Wu, P. Yang, Y. Xie, H. C. Woodruff, X. Rao, J. Guiot, A.-N. Frix, R. Louis, M. Moutschen, J. Li, J. Li, C. Yan, D. Du, S. Zhao, Y. Ding, B. Liu, W. Sun, F. Albarello, A. D'Abramo, V. Schininà, E. Nicastri, M. Occhipinti, G. Barisione, E. Barisione, I. Halilaj, P. Lovinfosse, X. Wang, J. Wu, and P. Lambin, "Development of a clinical decision support system for severity risk prediction and triage of covid-19 patients at hospital admission: an international multicenter study," *European Respiratory Journal*, 2020. [Online]. Available: https://erj.ersjournals.com/content/early/2020/06/25/13993003.01104-2020

[14] A. X. Garg, N. K. J. Adhikari, H. McDonald, M. P. Rosas-Arellano, P. J. Devereaux, J. Beyene, J. Sam, and R. B. Haynes, "Effects of Computerized Clinical Decision Support Systems on Practitioner Performance and Patient OutcomesA Systematic Review," *JAMA*, vol. 293, no. 10, pp. 1223–1238, 03 2005. [Online]. Available: https://doi.org/10.1001/jama.293.10.1223

[15] T. J. Bright, A. Wong, R. Dhurjati, E. Bristow, L. Bastian, R. R. Coeytaux, G. Samsa, V. Hasselblad, J. W. Williams, M. D. Musty, L. Wing, A. S. Kendrick, G. D. Sanders, and D. Lobach, "Effect of clinical decision-support systems: A systematic review," *Annals of internal medicine*, vol. 157, no. 1, pp. 29–43.

[16] T. Davenport and R. Kalakota, "The potential for artificial intelligence in healthcare," *Future healthcare journal*, vol. 6, no. 2, pp. 94–98, Jun 2019, 31363513[pmid]. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/31363513

[17] *Healthcare technology innovation adoption : electronic health records and other emerging health information technology innovations*, ser. Innovation, Technology, and Knowledge Management. Cham, Switzerland: Springer, 2016 - 2016.

[18] H. Petkus, J. Hoogewerf, and J. C. Wyatt, "What do senior physicians think about ai and clinical decision support systems: Quantitative and qualitative analysis of data from specialty societies," *Clinical Medicine*, vol. 20, no. 3, pp. 324–328, 2020. [Online]. Available: https://www.rcpjournals.org/content/20/3/324

[19] M. Laka, A. Milazzo, and T. Merlin, "Factors that impact the adoption of clinical decision support systems (cdss) for antibiotic management," *International journal of environmental research and public health*, vol. 18, no. 4, pp. 1901–, 2021.

[20] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[21] S. Berman, "Clinical decision making," in *Berman's Pediatric Decision Making (Fifth Edition)*, fifth edition ed., L. Bajaj, S. J. Hambidge, G. Kerby, and A.-C. Nyquist, Eds. Saint Louis: Mosby, 2011, pp. 1–6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780323054058000103

[22] D. L. Sackett, W. M. C. Rosenberg, J. A. M. Gray, R. B. Haynes, and W. S. Richardson, "Evidence based medicine: what it is and what it isn't," *BMJ*, vol. 312, no. 7023, pp. 71–72, 1996. [Online]. Available: https://www.bmj.com/content/312/7023/71

[23] B. Hernandez, P. Herrero, T. Rawson, L. Moore, E. Charani, A. Holmes, and P. Georgiou, "Data-driven web-based intelligent decision support system for infection management at point-of-care: Case-based reasoning benefits and limitations," 01 2017, pp. 119–127.

[24] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *Ai communications*, vol. 7, no. 1, pp. 39–59, 1994.

[25] D. Chushig-Muzo, C. Soguero-Ruiz, A. P. Engelbrecht, P. De Miguel Bohoyo, and I. Mora-Jiménez, "Data-driven visual characterization of patient health-status using electronic health records and self-organizing maps," *IEEE Access*, vol. 8, pp. 137 019–137 031, 2020.

[26] J. Li, Y. Peng, Y. Zhang, P. Zhang, Z. Liu, H. Lu, L. Peng, L. Zhu, H. Xue, Y. Zhao, X. Zeng, Y. Fei, and W. Zhang, "Identifying clinical subgroups in igg4-related disease patients using cluster analysis and igg4-rd composite score," *Arthritis Research & Therapy*, vol. 22, 01 2020.

[27] D. Ranti, A. J. Warburton, K. Hanss, D. Katz, J. Poeran, and C. Moucha, "K-means clustering to elucidate vulnerable subpopulations among medicare patients undergoing total joint arthroplasty," *The Journal of Arthroplasty*, vol. 35, no. 12, pp. 3488 – 3497, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0883540320307282

[28] M. Gordon, A. Moser, and E. Rubin, "Unsupervised analysis of classical biomedical markers: Robustness and medical relevance of patient clustering using bioinformatics tools," *PloS one*, vol. 7, p. e29578, 03 2012.

[29] M. Elbattah and O. Molloy, "Data-driven patient segmentation using k-means clustering: the case of hip fracture care in ireland," *Proceedings of the Australasian Computer Science Week Multiconference*, 2017.

[30] H. Estiri, J. G. Klann, and S. N. Murphy, "A clustering approach for detecting implausible observation values in electronic health records data," *BMC Medical Informatics and Decision Making*, vol. 19, no. 1, p. 142, Jul 2019. [Online]. Available: https://doi.org/10.1186/s12911-019-0852-6

[31] K. Magoev, V. V. Krzhizhanovskaya, and S. V. Kovalchuk, "Application of clustering methods for detecting critical acute coronary syndrome patients," *Procedia Computer Science*, vol. 136, pp. 370 – 379, 2018, 7th International Young Scientists Conference on Computational Science, YSC2018, 02-06 July2018, Heraklion, Greece. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050918315837

[32] J. Zhang and D. Chang, "Semi-supervised patient similarity clustering algorithm based on electronic medical records," *IEEE Access*, vol. 7, pp. 90 705–90 714, 2019.

[33] T. Ronan, Z. Qi, and K. Naegle, "Avoiding common pitfalls when clustering biological data," *Science Signaling*, vol. 9, pp. re6–re6, 06 2016.

[34] K. J. Cios, *Data Mining*, 1st ed. Springer US, 2007.

[35] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*, ser. The Morgan Kaufmann series in data management systems. Morgan Kaufmann, 2011.

[36] M. Kyan, *Unsupervised learning : a dynamic approach*, ser. IEEE series on computational intelligence. Hoboken, New Jersey : John Wiley & Sons, Inc., 2014.

[37] M. L. P. Bueno, A. Hommersom, P. J. F. Lucas, and J. Janzing, "A data-driven exploration of hypotheses on disease dynamics," in *Artificial Intelligence in Medicine*,

D. Riaño, S. Wilk, and A. ten Teije, Eds. Cham: Springer International Publishing, 2019, pp. 170–179.

[38] T. Hastie, R. Tibshirani, and J. Friedman, *Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer series in statistics. New York: Springer, 2009.

[39] D. Wang, "Unsupervised learning: Foundations of neural computation," *AI Magazine*, vol. 22, no. 2, p. 101, Jun. 2001. [Online]. Available: https://ojs.aaai.org/index.php/aimagazine/article/view/1565

[40] S. J. S. J. Russell, *Artificial intelligence : a modern approach*, global, third edition. ed., ser. Prentice Hall series in artificial intelligence. Boston: Pearson, 2016.

[41] S. Zhang, C. Zhang, and Q. Yang, "Data preparation for data mining," *Applied artificial intelligence*, vol. 17, no. 5-6, pp. 375–381, 2003.

[42] Y.-H. Hu, W.-C. Lin, C.-F. Tsai, S.-W. Ke, and C.-W. Chen, "An efficient data preprocessing approach for large scale medical data mining," *Technology and health care*, vol. 23, no. 2, pp. 153–160, 2015.

[43] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[44] B. Seijo-Pardo, A. Alonso-Betanzos, K. P. Bennett, V. Bolón-Canedo, J. Josse, M. Saeed, and I. Guyon, "Biases in feature selection with missing data," *Neurocomputing*, vol. 342, pp. 97 – 112, 2019, advances in artificial neural networks, machine learning and computational intelligence. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231219301493

[45] M. G. Pecht and M. Kang, *Machine Learning: Data Pre-processing.* Wiley, 2019, pp. 111–130.

[46] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing (Amsterdam)*, vol. 237, pp. 350–361, 2017.

[47] A. Zheng, *Feature engineering for machine learning : principles and techniques for data scientists*, 1st ed. O'Reilly Media, 2018.

[48] C. Li, "Data mining for direct marketing: Problems and solutions," 1999.

[49] R. Houari, A. Bounceur, M.-T. Kechadi, A.-K. Tari, and R. Euler, "Dimensionality reduction in data mining: A copula approach," *Expert systems with applications*, vol. 64, pp. 247–260, 2016.

[50] C. Boutsidis, A. Zouzias, M. W. Mahoney, and P. Drineas, "Randomized dimensionality reduction for k -means clustering," *IEEE Transactions on Information Theory*, vol. 61, no. 2, pp. 1045–1062, 2015.

[51] E. Alpaydin, *Introduction to machine learning*, 3rd ed., ser. Adaptive computation and machine learning. Cambridge, Massachusetts ; London, England : The MIT Press, 2014.

[52] I. T. Jollife and J. Cadima, "Principal component analysis: A review and recent developments," *Philosophical transactions of the Royal Society of London. Series A: Mathematical, physical, and engineering sciences*, vol. 374, no. 2065, pp. 20 150 202–20 150 202, 2016.

[53] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[54] E. Oliver, I. Vallés-Perez, R.-M. Baños, A. Cebolla, C. Botella, and E. Soria-Olivas, "Visual data mining with self-organizing maps for "self-monitoring" data analysis," *Sociological methods & research*, vol. 47, no. 3, pp. 492–506, 2018.

[55] N. Manukyan, M. J. Eppstein, and D. M. Rizzo, "Data-driven cluster reinforcement and visualization in sparsely-matched self-organizing maps," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 5, pp. 846–852, 2012.

[56] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[57] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *CoRR*, vol. abs/1906.02691, 2019. [Online]. Available: http://arxiv.org/abs/1906.02691

[58] G. Gan, *Data clustering theory, algorithms, and applications*, ser. ASA-SIAM series on statistics and applied probability ; 20. Philadelphia, Pa.: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2007.

[59] R. Xu and D. Wunsch, *Clustering*, 1st ed., ser. IEEE Press Series on Computational Intelligence. Hoboken: Wiley-IEEE Press, 2008, vol. 10.

[60] B. Everitt, "Cluster analysis," *Quality & quantity*, vol. 14, no. 1, pp. 75–100, 1980.

[61] B. S. Everitt, S. Landau, M. Leese, and D. Stahl, *Cluster Analysis*, 5th ed., ser. Wiley series in probability and statistics. New York: Wiley, 2010.

[62] O. Maimon and L. Rokach, *Data Mining and Knowledge Discovery Handbook*, 1st ed. New York, NY: Springer Science + Business Media, 2005.

[63] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD’96. AAAI Press, 1996, p. 226–231.

[64] “The oucru datasets¶.” [Online]. Available: https://bahp.github.io/vital-oucru-clinical/datasets/overview.html

[65] T. pandas development team, “pandas-dev/pandas: Pandas,” Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[66] J. Peng, W. Wu, B. Lockhart, S. Bian, J. N. Yan, L. Xu, Z. Chi, J. M. Rzeszotarski, and J. Wang, “Dataprep.eda: Task-centric exploratory data analysis for statistical modeling in python,” in *Proceedings of the 2021 International Conference on Management of Data (SIGMOD ’21), June 20–25, 2021, Virtual Event, China*, 2021.

[67] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[68] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[69] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[70] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. Seljebotn, and K. Smith, “Cython: The best of both worlds,” *Computing in Science Engineering*, vol. 13, no. 2, pp. 31 –39, 2011.

[71] G. Vettigli, “Minisom: minimalistic and numpy-based implementation of the self organizing map,” 2018. [Online]. Available: https://github.com/JustGlowing/minisom/

[72] P. T. Inc. (2015) Collaborative data science. Montreal, QC. [Online]. Available: https://plot.ly

[73] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Commun. ACM*, vol. 18, no. 9, p. 509–517, Sep. 1975. [Online]. Available: https://doi.org/10.1145/361002.361007

[74] H. M. Kakde, “Range searching using kd tree.” Aug 2005. [Online]. Available: http://www.cs.utah.edu/~lifeifei/cis5930/kdtree.pdf

[75] “Shepard diagram.” [Online]. Available: https://www.oxfordreference.com/view/10.1093/oi/authority.20110803100501161

[76] J. Duignan, “Pearson correlation,” 2016. [Online]. Available: https://www.oxfordreference.com/view/10.1093/acref/9780191792236.001.0001/acref-9780191792236-e-423

[77] M. Elliot, I. Fairweather, W. Olsen, and M. Pampaka, “Spearman’s correlation,” 2016. [Online]. Available: https://www.oxfordreference.com/view/10.1093/acref/9780191816826.001.0001/acref-9780191816826-e-0382

[78] W. J. Krzanowski, *Principles of multivariate analysis : a user’s perspective*, revised edition. ed., ser. Oxford statistical science series ; 23. Oxford: Oxford University Press, 2008.

[79] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional space,” in *Database Theory — ICDT 2001*, J. Van den Bussche and V. Vianu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 420–434.

[80] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is ”nearest neighbor” meaningful?” *ICDT 1999. LNCS*, vol. 1540, 12 1997.

[81] J. Grus, *Data science from scratch*, 1st ed. Sebastopol, CA: O’Reilly, 2015 - 2015.

[82] J. Whitehorn and J. Farrar, “Dengue,” *British medical bulletin*, vol. 95, no. 1, pp. 161–173, 2010.

[83] M. Wattenberg, F. Viégas, and I. Johnson, “How to use t-sne effectively,” *Distill*, 2016. [Online]. Available: http://distill.pub/2016/misread-tsne

[84] J. Tian, M. Azarian, and M. Pecht, “Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm,” in *European Conference of the Prognostics and Health Management Society 5*, 2014.

[85] B. C. Boehmke and B. M. Greenwell, “Hands-on machine learning with r,” 2019.

[86] W. H. Organization, “Dengue haemorrhagic fever : diagnosis, treatment, prevention and control,” pp. Chinese–1ed–, 1997.

[87] J. W. Palmer, "Web site usability, design, and performance metrics," *Information systems research*, vol. 13, no. 2, pp. 151–167, 2002.

[88] A. N. Tuch, J. A. Bargas-Avila, K. Opwis, and F. H. Wilhelm, "Visual complexity of websites: Effects on users' experience, physiology, performance, and memory," *International journal of human-computer studies*, vol. 67, no. 9, pp. 703–715, 2009.

[89] A. Dickinger and B. Stangl, "Website performance and behavioral consequences: A formative measurement approach," *Journal of business research*, vol. 66, no. 6, pp. 771–777, 2013.

[90] O. Hagai. (2020, Feb) Need for speed: Top 10 web performance metrics you must monitor. [Online]. Available: https://www.namogoo.com/blog/conversion-rate-optimization/top-10-web-performance-metrics/

[91] E. Frieman. (2019, Aug) Website performance metrics. [Online]. Available: https://keymedium.com/website-performance-metrics/

[92] J. Nielsen, *Usability Engineering.* Morgan Kaufmann, 1994.

Appendix A

Project source code

The project repository is available on Github:

`https://github.com/ostiff/fyp2020-oss1017`