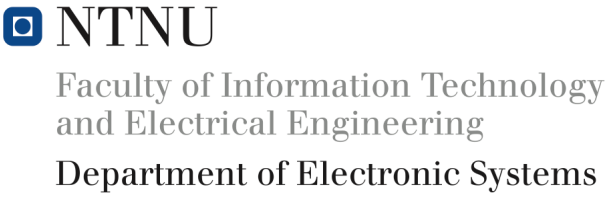
Auto-strain master thesis

Written by
Yohann Jacob Sandvik

Master thesis EMNEKODE

Supervised by
Lasse Løvstakken



Department of electronic systems
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology
June 25, 2020

Abstract

This is the abstract.

Acknowledgements

These are my acknowledgements.

Contents

List of Abbreviations

2CH 2-Chamber. 7, 41, 48–50, 57, 61

4CH 4-Chamber. 7, 13, 39–41, 54, 59, 61

ANN Artificial Neural Network. 6–9, 13, 19–22, 24, 37, 42–44, 54, 56, 63, 68

APLAX Apical-Long-Axis. 41, 61

ARI Adjusted Rand Index. 8, 9, 23, 24, 46, 47, 49, 51, 52, 58, 60, 61, 67

BMI Body Mass Index. 6, 26, 27

CNN Convolutional Neural Network. 21

DOR Diagnostic Odds' Ratio. 7–9, 23, 43, 46, 48–52, 54–58, 60, 61, 63–68

DTW Dynamic Time Warping. 6, 15, 16, 24, 39, 41

EF Ejection Fracture. 6, 27, 28, 30, 31, 37, 50, 52, 55, 56, 60

FN False Negatives. 8, 9, 22, 23, 41, 43, 46, 61, 63, 65

FP False Positives. 8, 9, 22, 23, 41, 43, 46, 65

GLS Global Longitudinal Strain. 6, 7, 13, 27, 30–33, 36, 37, 39–42, 49, 50, 52–54, 56–58, 61, 62

GP Gaussian Process. 44, 45

KNN K Nearest Neighbors. 44

LCM Local Cost Matrix. 6, 15, 16

LSTM Long Short-term Memory. 21

ML Machine Learning. 8, 36, 55

MLP Multi-layer Perceptron. 6, 19, 44

PVC Peak-value Clustering. 7, 14, 15, 37, 38, 41

PVSC Peak-value Supervised Classifier. 37, 44

RBF Radial Basis Function. 44, 45

ReLu Rectified Linear Unit. 19, 44

RLS Regional Longitudinal Strain. 6, 34–37, 42, 54

RNN Recurrent Neural Network. 21

SGD Stochastic Gradient Descent. 20, 22, 24, 43, 44

SVC Support Vector Classifier. 44

TN True Negatives. 8, 9, 22, 23, 41, 43, 46, 58, 63, 65

TP True Positives. 8, 9, 22, 23, 41, 43, 46, 58, 65

TSC Time-series Clustering. 6–9, 13–15, 24, 37–39, 41, 48, 49, 56–58, 65–68

List of Figures

List of Tables

Chapter 1

Introduction

This is the introduction.

1.1 Motivation

This will be the section on the motivation for the assignment.

1.2 Objective

This will be the section where i outline the objective of the assignment.

Objectives

1. Can an ML model be used to identify the following three binary target variables, using longitudinal strain from the left ventricle of the heart? Does the patient have heart failure, does the patient have a heart disease, are the individual segments of the patients left ventrice acting abnormally.
2. Which type of machine learning is best suited for predicting the aforementioned target variables, supervised, or unsupervised learning models?
3. Which type of dataset works best for a ML model to predict the target variables, a dataset consisting of longitudinal strain curves, or a dataset that consists of peak systolic longitudinal strain values in combination with EF?

1.3 Structure of Thesis

Here the outline for the rest of the assignment will be given.

Chapter 2

Myocardial Imaging and Echocardiography

This will be a kind of theory section about echocardiography, and strain imaging.

2.1 Basic Cardiology

The heart is an autonomous muscle that is responsible for pumping oxygenated blood from the lungs, into the rest of the body and pumping unoxegenated blood from the rest of the body, into the lungs. The heart can be divided into four separate chambers: The right atrium, the left atrium, the right ventricle and the left ventricle. The atriums are responsible for pumping unoxegenated blood into the lungs, and the ventricles are responsible for pumping oxygenated blood into the body. One heart cycle is the time period it takes the heart muscles to make a full contraction and relaxation. The period of the heart cycle where the heart relaxes, and fills with blood is called the *diastole*, and the period of the heart cycle when the heart contracts and pumps blood throughout the body is called the *systole*. Cardiology is the branch of medicine that deals with the heart, and parts of the vascular system [1]. Cardiologists are doctors that specialize in the field of cardiology.

2.2 Introduction to Myocardial Imaging

2.3 Myocardial Strain Estimation

2.4 Heart Diseases

2.5 Chapter Summary

Chapter 3

Machine Learning Theory

This section will act as a theory section for the machine learning models used. A machine learning models are a subset of artificial intelligence models. Machine learning models extract rules from data, which can then be applied to classify, or estimate components of another dataset called the *target variable*. What makes mahine learning models different from other artificial intelligence models, is that the rules for making predictions on the target variable are not given to the model explicitly. Instead, the models are given data and extract the rules for making predictions themselves. Machine learning algorithms are formally divided into *supervised learning*, *unsupervised learning* and *semi-supervised learning*. Supervised learning models require labelled datasets to extract information from the dataset, and are usually used to perform classification tasks, or to estimate a variable that is considered dependent on the input variables (regression). Unsupervised learning algorithms do not require labelled datasets. There also exist hybrid models called *semi-supervised learning* that use a combination of labelled and unlabelled datasets. The two sections of this chapter are dedicated to the two most central machine learning models encountered in this work; clustering and ANN. The clustering section will give the theoretical background of the similarity measures, and clustering algorithm used, and the deep neural networks section will explain the basic building blocks of ANN, the different layers in a ANN, and how they are trained. In the paragraph below the definition of a time series is given, which is the definition that is used throughout this work.

What Is a Time Series?

A time series is defined as a set of observations $\{x_t\}$ recorded at a specific time t. A discrete time series is a time series where the set of times when observations are made (T_0) is discrete [2]. A multivariate time series can be viewed as a set of vectors $\{\mathbf{x}_t\}$ where each set of vector elements $\{x_t^i\}$ is an individual time series. This means that the elements of the same vector $[x_t^1, x_t^2, ..., x_t^N]$ are separate observations. A GLS curve extracted from an ultrasound video from the 4CH view of a patient can be considered a univariate time series, while the GLS curves extracted from the ultrasound videos of all the three views for a single patient can be considered a multivariate time series.

3.1 Clustering

There are three types of TSC, *whole-series TSC*, *subsequence TSC* and *time-point TSC*. Whole-series TSC is when multiple "whole" time series are clustered with respect to each other. Subsequence TSC comprises the clustering of subsequences of the same time series with respect to

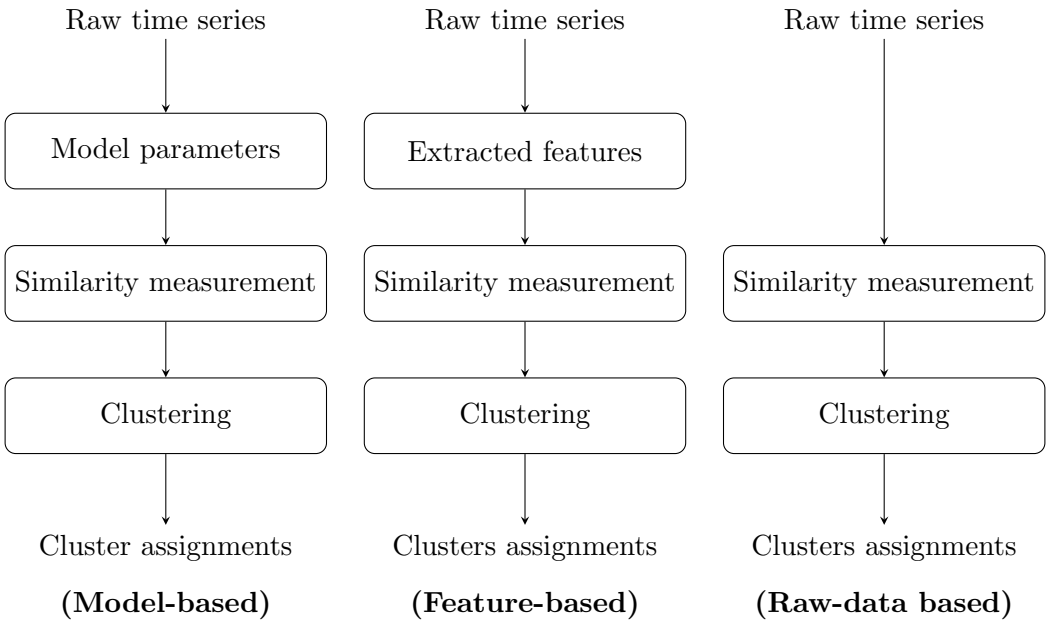
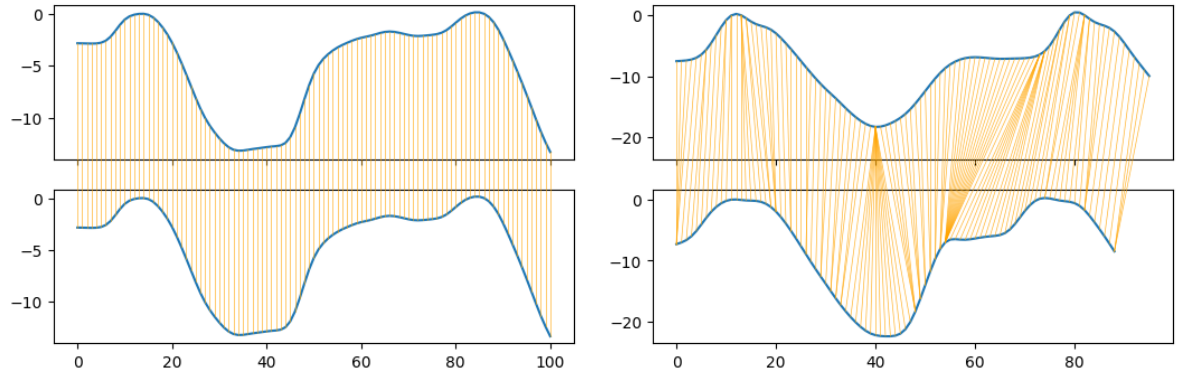


Figure 3.1: Illustration of the three approaches to whole-series TSC, and their components. The illustration is inspired by figure 2 in Aghabozorgi, Shirkhorshidi, and Wah [3].

each other. The defining difference between whole-series and subsequence TSC is that whole-series TSC clusters multiple time series while subsequence TSC clusters different subsequences of the same time series. When performing time-point TSC the goal is to cluster individual observations of a time series with regard to to each other. In this review we will only consider work using whole-series TSC, so when the phrase *time-series clustering* is used, one can assume that whole-series TSC is what is being refered to.

Whole series TSC can broadly be divided into three main approaches. The raw-data based approach, the feature-based approach and the model based approach. In the raw-data based approach one measures the similarity between the raw time series themselves and clusters them based on this. When clustering raw time series the majority of the work goes into selection of similarity metric and clustering algorithm, and one clusters the time series with regard to similarity in time or similarity in shape [3]. In the feature-based approach one also clusters time series with regard to similarity in time, and shape, but the work is somewhat shifted away from choice of similarity metric and over to choice of representation. Either to extract more relevant information from the time series, or to reduce the computational complexity of the similarity measurement. In the model-based approach the goal is most often to cluster time series with regard to the underlying data generating process [4]. The underlying assumption being that two time series that appear different might still have been generated by the same process.
The common denominator of the three approaches to TSC mentioned is that they are all made up of three distinct parts: representation method, similarity measurement and clustering algorithm. This is illustrated in figure 3.1. Another key aspect of the TSC model is what the **objective** is. There are broadly regarded to be three objectives, one can cluster with regard to: Similarity in time, similarity in shape and similarity in change CITATION. When calculating the similarity between all combinations of time series, the resulting similarity metric are stored in what is called a *dissimilarity matrix*. The choice of similarity metric is important in a raw-data approach as it decides which aspects of the time series will be used to measure (dis)similarity, and has a has a significant impact on the time-complexity of the clustering system. PVC has a similar approach as raw-data based TSC, the dissimilarity between data points are measured which are



(a) Sample-wise Euclidean distance.

(b) DTW distance.

Figure 3.2: An illustration of the difference between sample-wise Euclidean distance between time series, and DTW distance between time series.

passed on the clustering algorithm. In the subsection below, the dissimilarity measures used in the clustering models of this work are described.

3.1.1 Dissimilarity metric

When clustering point-values, the choice of metric used to measure dissimilarity between the data objects are usually some sort of distance measure. The choices of distance measures are varied and plentiful. Options include: Euclidean distance, Manhatten distance and Minkowski distance. In the PVC models the Euclidean distance is used because it is the most easy to interpret geometrically. It is defined in equation (3.1) for two data objects x, and y of N dimensions.

$$ED(x, y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} \tag{3.1}$$

In the raw-data based approach to TSC, choice of dissimilarity metric is paramount, and is chosen based on what objective of the TSC is, and the different lengths of the time series to be compared. When clustering with regard to similarity in shape, the similarity metric can be lock-step (one-to-one) or elastic (one-to-many) [3]. An example of a lock-step measure is the use of Euclidean distance to measure the distance between time series sample-wise. However, this becomes problematic when the time series are not of equal length. Dynamic Time Warping (DTW) distance is a powerful alternative for Euclidean distance to measure the shape-based distance between two time series. To understand how the DTW distance works as a dissimilarity metric, one can imagine that it warps one time series such that the two series are equal in length, and then measures the Euclidean distance between them. This is illustrated in figure 3.2. DTW is probably most famous from speech recognition where it is applied to find out which phoneme[1] in a dictionary of phonemes is the optimal fit to a recorded sound. To calculate the DTW distance between two time series x and y of length n and m respectively. First an $(n \times m)$ matrix is constructed called the Local Cost Matrix (LCM). Element LCM(i, j) is the sample-wise quadratic distance between x_i and y_i $((x_i - y_i)^2)$. The next step is to create a warping path $P = \{p_1, p_2, ..., p_L\}$ across the LCM. The warping path must fulfill three conditions: the boundary condition, the continuity condition and the monotonicity condition.

[1]Phoneme is a term from speech recognition, and refers to the biggest unit of sound for which the frequency spectrum is constant. phonemes are considered as the "atomic sounds" that make up speech.

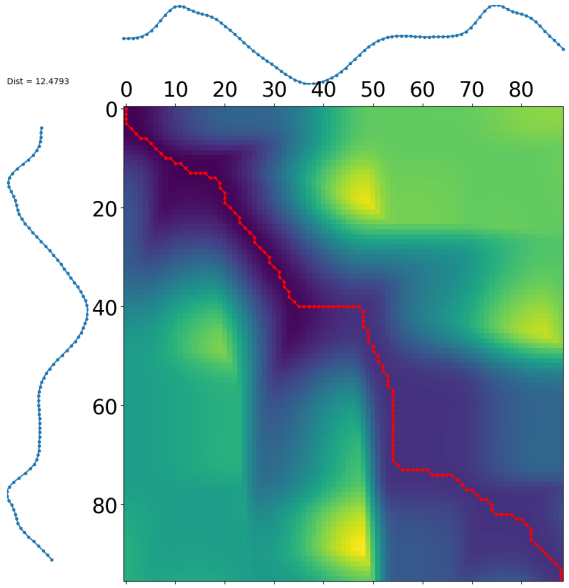


Figure 3.3: An illustration of DTW distance. The big coloured square is the LCM, each monochromatic subsquare in is an entry in the LCM. The color of each subsquare indicates the magnitude of the quadratic distance in that entry, blue indicates low, and green and yellow indicate higher values. The red line is the warping path.

1. **Boundary**: The path must begin, and end in the corners of the LCM. $p_0 = \text{LCM}(1,1)$, $p_L = \text{LCM}(\text{n},\text{m})$
2. **Continuity**: Two adjacent warping steps p_k and p_{k+1} must be equal to adjacent elements on the LCM. This means that the matrix elements p_k and p_{k+1} are equal to must be adjacent horizontally, vertically or diagonally.
3. **Monotonicity**: The warping path must increase monotonically. This means that the warping path cannot go backwards index-wise. If one combines the continuity, and monotonicity constraints, and lets $p_k = \text{LCM}(i,j)$, valid values for p_k are $\text{LCM}(i+1,j)$, $\text{LCM}(i,j+1)$ and $\text{LCM}(i+1,j+1)$.

The warping distance of the warping path P is the sum of the LCM elements that entries of P are equal to. The DTW distance between time series x and y is then defined as the square root of the smalles possible warping distance between x and y. The warping path corresponding to the smallest warping distance can be found by using a reccurrent algorithm from dynamic programming shown in equation (3.2) CITATION PJOTR ELLEFSEN.

$$\begin{aligned} p_1 &= \text{LCM}\{1,1\}, p_L = \text{LCM}\{n,m\} \\ p_i &= \text{LCM}\{f,g\} \\ p_{i+1} &= \min\{\text{LCM}\{f+1,g\}, \text{LCM}\{f,g+1\}, \text{LCM}\{f+1,g+1\}\} \end{aligned} \tag{3.2}$$

Although the DTW distance is more flexible than estimating Euclidean distance between two time series, it comes at the cost of much higher run time and space requirements. The time complexity for calculating the dissimilarity matrix of a set of N time series using the DTW distance is $O\left(nmN^2\right)$ CITATION? An illustration of how the DTW distance between two time series is estimated is shown in figure 3.3.

3.1.2 Agglomorative Hierarchical Clustering

The agglomorative hierarchical clustering algorithm is the chosen clustering algorithm in this work. It is a *hard* clustering algorithm meaning that data objects are given a single cluster assignment, and do not have partial memberships to many different clusters. Clustering algorithms that assign data objects partial memberships to many clusters are called *soft* clustering algorithms.

Partitional clustering algorithms is a family of clustering algorithms that are an alternative to the family of hierarchical clustering algorithms. Partitional methods work iteratively and rely on defining prototypes that represent the cluster centre. In the first iteration the prototypes are randomly initialized, then the dissimilarity between all data objects and the prototypes are calculated, the data objects are then assigned to the cluster where the dissimilarity to the cluster prototype is minimal. The final step is to update the cluster prototypes such that they best represent the center of the new cluster. These steps repeat untill the value of the cluster prototypes, and cluster membership assignments converge.

Hierarchical clustering algorithms have have two central advantages over partitional clustering algorithms, such as K-means, K-medoids and fuzzy C-means. The first advantage is that the user does not have to decide the number of clusters they want to partition the dataset into prior to using the algorithm, the second is that due to the reliance of cluster prototypes, and their random initialization, the cluster assingments yielded when the partitional algorithm are non-deterministic. The clusters assignments that a partitional algorithm converges to is dependent on what values the cluster prototypes are given upon initialization. In fact, CITE ESPEN found that one is not guaranteed convergence at all. Hierarchical clustering algorithms will always yield the same cluster assignments, given that the same dissimilarity matrix is inputed.

There are two main types of hierarchical clustering algorithms, *divisive* and *agglomorative*. To understand the difference between these two algorithms, it helps to first understand how agglomorative hierarchical clustering works. Assume one is applying the hierarchical clustering algorithm to cluster a dataset of N data objects. In the initial step, the algorithm takes the dissimilarity matrix as input and every data object in the dataset is regarded as a separate cluster. Next, the case of $N-1$ clusters is considered, two of the existing clusters are merged based on which clusters have the lowest dissimilarity such that there then are $N-1$ clusters. Dissimilarity between clusters is estimated with what is called a *linkage criterion*, which will be expanded upon later. This step of merging existing clusters is repeated until all data objects are contained in one cluster. The result is then a hierarchy of clusters called a *dendrogram*, that can yield cluster assignments at all the possible number of clusters. If one says that agglomorative hierarchical clustering has a bottom-top approach, divisive hierarchical clustering can be said to have a top-bottom approach. It starts at the top of the dendrogram with all data objects in one cluster, and continously splits the cluster until every object is contained in its own cluster. Of the two types of hierarchical clustering the agglomorative approach is the most popular CITATION. What needs to be expanded upon the different types of linkage criterion. In this work seven different linkage criterion are used, which are detailed below.

- **Single linkage**: Computes the dissimilarity between two clusters as the smallest dissimilarity between two individual members of each cluster [5].
- **Complete linkage**: Computes the dissimilarity between two clusters as the biggest dissimilarity between two individual members of each cluster [6].
- **Average linkage**: Computes the dissimilarity between two clusters as the average dissimilarity between all members of each cluster [5].

- **Ward linkage**: Computes the dissimilarity between two clusters as the increase in sum squared dissimilarity of the entire cluster that would be the result of merging the two clusters [7].
- **Centroid linkage**: Computes the dissimilarity between clusters by representing each cluster with a "centroid" which is another word for a cluster prototype. Dissimilarity between clusters is then computed as dissimilarity between the centroids of each cluster. After the two clusters are merged a new centroid is computed based on all the cluster members of the two clusters merged. CITE SCIPY-CLUSTER-HIERARCHY-LINKAGE DOCUMENTATION.
- **Median linkage**: Computes dissimilarity between two clusters in the same way as the centroid linkage, the only difference being that after the clusters are merged, the new centroid is computed as the average of the two previous centroids. CITE SCIPY-CLUSTER-HIERARCHY-LINKAGE DOCUMENTATION.
- **Weighted linkage**: Works in a method similar to to the average linkage, the only difference being that after two clusters are merged, this linkage requires all the entries in the dissimilarity matrix that pertain to members of this cluster to be averaged. This reduces the number of computations required further down the line because there will be fewer dissimilarity values to average. CITE SCIPY-CLUSTER-HIERARCHY-LINKAGE DOCUMENTATION.

One of the obvious disadvantages of the hierarchical clustering algorithms is that they have quadratic time complexity $O(N^2)$, they have also received critique for lacking flexibility [3]. The lack of flexibility is due to the fact that after two clusters are merged they cannot be split, for re-evaluation when lower number of clusters are considered.

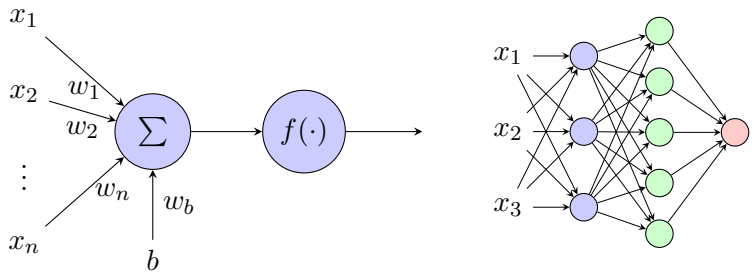
3.1.3 Curse of Dimensionality

3.2 Deep Neural Networks

This section will explain how layers of perceptrons form an artificial neural network, how said network is trained, the function of convolutional layers, and recurrent layers in an ANN, and will discuss the challenges of underfitting and overfitting.

3.2.1 Multi-layer perceptrons

Figure 3.4a depicts the building blocks of an ANN, the perceptron. The perceptron is a model of an artificial neuron, it takes in n inputs, performs a weighted sum of the inputs and a bias b, and sends the sum through what is called an activation function. A single perceptron is only able to perform binary classification on points that are linearly separable. However, by combining multiple perceptrons into a layer, and multiple layers of perceptrons into a Multi-layer Perceptron (MLP), they are able to capture complex non-linear relationships.

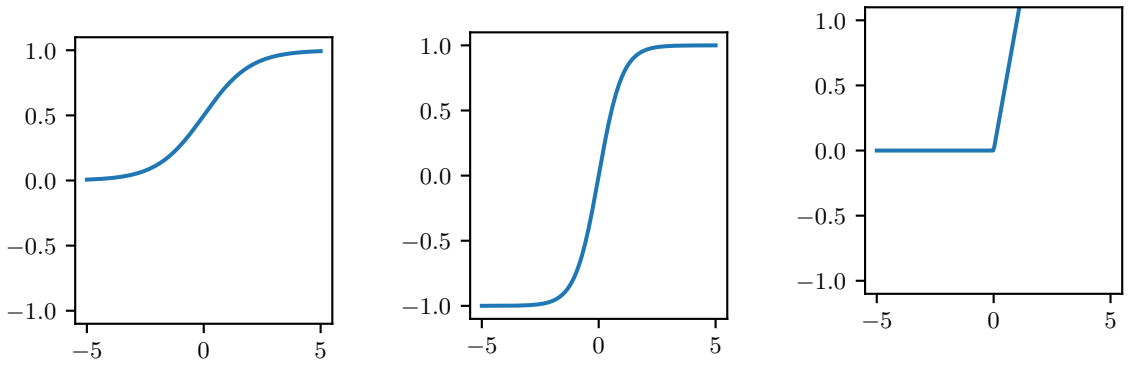


(a) (b)

Figure 3.4: (a) A Perceptron. (b) Example of a simple ANN known as an MLP.

$$O(\mathbf{x}) = f\left(\sum_{i=1}^{N} w_i x_i + w_b b\right) \tag{3.3}$$

Equation (3.3) shows what the output of a perceptron ($O(\mathbf{x})$) is in terms of its weights $\{w_i\}$, the input $\mathbf{x}$, its activation function $f(\cdot)$ and its bias b. The purpose of an activation function is to give each perceptron the ability to perform actions that are not purely linear on its inputs [8]. Consider that the absense of an activation function, all a perceptron is doing is outputing a weighted sum of its inputs. Any continous function can in principle be an activation function, but there are some functions that are more popular than others. Figure 3.5 shows the three most popular activation functions used in modern ANN, the sigmoid function, tanh function, and Rectified Linear Unit (ReLu). The sigmoid function was one of the first activation functions introduced, and it shares many similar characteristics with the tanh function. The sigmoid, and tanh functions are both hyperbolic functions that grant non-linear properties to the perceptron. However, the ReLu function is often preferred over the two former functions for two important reasons. First, the hyperbolic functions suffer an issue of saturation when the weighted sums of the input becomes sufficiantly large, the ReLu does not. The second reason is not as technical, but is still important. Since an ANN can be made up o hundreds, or maybe even thousands of perceptrons, the computation of complex exponential functions for every unit is computationally expensive, whereas the computation of the ReLu is significantly less expensive.



(a) sigmoid$(x) = \frac{1}{1+e^{-x}}$ (b) $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ (c) ReLu$(x) = \begin{cases} x | x > 0 \\ 0 | x \leq 0 \end{cases}$

Figure 3.5: An illustration of the three most popular activation functions used for perceptrons in ANN.

3.2.2 Training

A simple ANN is depicted in figure 3.4b.The first layer in an ANN is called the input layer, the last layer is called the output layer and all the layers inbetween are called hidden layers. When an ANN makes a prediction it does what is called a *feed-forward computation*. The data is passed through the input layer, and sendt through the hidden layers, and finally through the output layer. When training an ANN one defines a loss function, $L(\theta)$ which estimates the error in the prediction as a function of the parameters of the ANN, θ. After a prediction is made with a feed-forward computation, and the error in the prediction is calculated using the loss function the trainable parameters of the network need to be updated. This updating of the weights can be considered a gradient optimization problem, and is solved using an algorithm called Stochastic Gradient Descent (SGD) [8]. The SGD algorithm is shown in equation (3.4), where l_r is called the *learning rate*.

$$\theta_{new} = \theta_{old} - l_r \frac{\partial L(\theta_{old})}{\partial \theta_{old}} \tag{3.4}$$

The estimation of the partial derivatives of the loss function with regard to the individual parameters of the ANN is a significant task, and is estimated with the back-propagation algorithm. The back-propagation algorithm estimates the partial derivatives of the loss function with regard to the network parameters by beginning with the output layer, and working its way backward through the hidden layers. It is given by the chain-rule of differentiation that since the output of hidden layer N in an ANN is a function of the layers preceding it, the partial derivative of a parameter in N will be dependent on the partial derivatives of all the layers coming after it. The computation of the back-propagation is expensive in terms of time and space, and is often computed on a GPU since it has many small cores and is capable of computing the same instruction on many data points. A challenge in training of ANN is the choice of l_r, if it is two small the model learns too slowly, and if it is too big one risks the possibility of overcompensating and increasing the error. Additionally, when parameters are getting close to values which correspond to a minimum of the loss function the gradients of the loss function tend to become vanishingly small [8]. To address these challenges, one often uses a *gradient descent optimizer* which changes the learning rate during training if overcompensation is detected, or if the gradients returned by the back-propagation algorithm become very small. One of the most common gradient descent optimizer used is called ADAM, but an explanation of the inner workings of ADAM falls outside the scope of this work. There exist alternatives to the SGD algorithm, such as batch gradient descent, and mini-batch gradient decent, but

Figure 3.6: An illustration of how a one dimensional convolutional layer works. The blue circles represent the input to the convolutional layer, the red circles represent units that make up the convolutional layer, the green circles represent the output of the convolutional layer, the thin arrows between units represent the weighted sum and the thick arrow represents the sliding of the filter over the input.

they will not be applied in this work. The term *epoch* or *training epoch* refers to the process of training the ANN model on the entire training set once. It is normal to train an ANN for multiple epochs, where the number of epochs depends on the complexity of the architecture.

3.2.3 Convolutional Layers

Layers of perceptrons where all the outputs of the previous layer are connected to all the inputs of the current layer are refered to as *dense*, and they are only one of many possible layers that can make up an ANN. Convolutional layers get their name from the convolution operator, and for time series they can viewed as a set one dimensional filters. Each sample in the filtered output is a weighted sum, passed through activation functions of a close neighborhood of samples of the input of the convolutional layer. This is illustrated in figure 3.6. A convolutional layer may apply multiple filters, which each produce a separate output. Convolutional layers are very popular in ANN used for computer vision tasks, because they can be used for detecting distinct features such as lines, and edges [8]. For time series, the features that are extracted could be linear regions, exponential regions or zero gradient regions. As the network gets deeper the features extracted by convolutional layers are combined to detect more complex structures such as periodicity in time-series data or furniture in an image. ANN that apply convolutional layers and dense layers are called a Convolutional Neural Network (CNN).

3.2.4 Recurrent Layers

An attribute that was long sought after was the ability of an ANN to detect time-dependent relations between the input. Especially in the fields of time-series analysis, natural language processing and video analysis. To address this problem special perceptrons with "memory" were introduced, and layers of these perceptrons are called *recurrent layers*. The way that the memory attribute was added to recurrent units, was by introducing a feedback loop such that the past output was added to the weighted sum of inputs with its own separate weight, as illustrated in figure 3.7. One implementation of this type of unit is called the Long Short-term Memory (LSTM) unit. It works by giving a memory unit to a perceptron which has three *gates* that regulate the flow of information within the unit: write, read, and flush. The write gate controls to which extent new inputs are allowed into the unit, the write gate controls the weighting that the old values in the unit are given, when calculating the output and the flush gate controls how long a particular value is allowed to remain in the unit [9]. ANN that apply recurrent layers and dense layers are called a Recurrent Neural Network (RNN). The name "deep neural network" is used for architectures that are "deep", meaning that they consist of many layers. There is no formal limit of how many layers an architecture must have before it is considered deep, so Artificial Neural Network (ANN) is the nomenclature that will be used about the model used in this work.

	Patient is sick	Patient is healthy
Patient tests sick	TP	FP
Patiet tests healthy	FN	TN

Table 3.1: Illustration of how the metrics TP, TN, False Positives (FP) and False Negatives (FN) are defined.

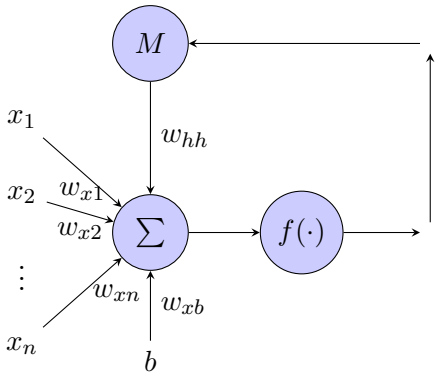


Figure 3.7: An simplified illustration of the memory in an LSTM unit.

3.2.5 Underfitting and Overfitting

When training an ANN for deployment it is common to divide the data one has at hand into a training set, a validation set and a test set. The training and validation sets are used during training, and the test set is used after training to benchmark the model. When an ANN is trained, SGD with back-propagation is performed on the training set, simultaneusly the model is evaluated on independent validation set. This is done to determine whether the model is *overfitting* on the training set. Overfitting is described as when a model performs well on the training set, but underperforms on the validation set, and test set. This is common among complex ANN architectures, and can be a sign that the architecture applied is too complex for the dataset. *Underfitting* is said to occur when the accuracy of the model on the training set is lower than to be expected, this is often a sign that architecture of the ANN is too simple, and can be expanded.

3.3 Evaluation metrics

In medicine it one often assesses a test for a specific disease in terms of how many True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) the test attains. The meaning of these terms are illustrated in table 3.1. If a patient is sick, and the test classifies the patient as sick this result is regarded as a True Positives, if a patient is sick, but is classified healthy by the test it is regarded as a False Negatives, if a patient is healthy, and is classified healthy by the test it is regarded as a True Negatives and if a patient is healthy, but is classified as sick this is regarded as a False Positives. These metrics are also used frequently for assessing the performance of binary classifiers. They can be used on multi-class classifiers as well, but then one would have to calculate a set of metrics for each class. For classifiers the aim is always to maximize the number of TP, and TN and minimize the number of FP, and FN. The common metric accuracy can be defined in terms of these metrics as $(\mathrm{TP}+\mathrm{TN})/(\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN})$.

3.3.1 Sensitivity, Specificity and Diagnostic Odds Ratio

Usually it can be helpful to combine the four metrics shown in table 3.1 into two more compact metrics known as sensitivity (true positive rate) and specificity (true negative rate), which are defined in equation (3.5). Sensitivity is defined as the number of positive cases correctly classified, divided by the total number of positive cases in the dataset. Similarily specificity is defined as the total number of negatives correctly classified divided by the total number of negatives in the dataset. If a dataset does not have an even distribution of positives or negatives the accuracy can be inflated if a model is only able to perform well at classifying one class. Analyzing the sensitivity and specificity separately allows one to get a better understanding of how well a model works at detecting each category. As with accuracy, sensitivity and specificity can range from zero to one.

$$\begin{aligned} \text{Sensitivity} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\ \text{Specificity} &= \frac{\text{TN}}{\text{TN} + \text{FP}} \end{aligned} \tag{3.5}$$

A third metric that is especially usefull when comparing multiple classifiers is known as Diagnostic Odds' Ratio (DOR). It is defined by equation (3.6). What one can see quickly is that the value of the DOR is unbounded, in contrast to the acccuracy, sensitivity and specificity metrics. This is both a blessing, and a curse. The advantage of this is that differences that may seem very small in terms of accuracy, sensitivity and specificity become very evident in the DOR, the disadvantage is that the metric becomes undefined if either FP, or FN are zero. An advantage of the DOR is that it takes both TP and TN into account, wheras other popular metrics such as the F1-score does not take TN into account.

$$\text{DOR} = \frac{\text{TP} \times \text{TN}}{\text{FP} \times \text{FN}} \tag{3.6}$$

3.3.2 Adjusted Rand Index

The Adjusted Rand Index (ARI) is a version of the Rand Index, that is "adjusted for chance". The Rand Index applied in binary classification problems is equivilant to accuracy [10]. However, it might be more helpful to view the ARI as a measure of how much the distribution of two groupings[2] of a dataset overlap. Given that one has a dataset X with n objects $X = \{x_1, x_2, ..., x_n\}$, and two groupings of this dataset, Y which has p different labels, and Z which has q different labels. The first step of estimating the DOR is setting up a contingency table shown in table 3.2 [10]. Here entry n_{ij} is the number of data objects that have label Z_i in the Z-grouping and label Y_j in the Y-grouping, a_i is the number of data objects with the Z_i label, and b_j is the number of data objects with the Y_j label. The ARI is then calculated according to equation (3.7)

$$\text{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \frac{\left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right]}{\binom{n}{2}}}{0.5 \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \frac{\left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right]}{\binom{n}{2}}} \tag{3.7}$$

[2]Groupings refers to a segregation of a dataset into distinct non-overlapping groups with separate labels. An example of a grouping can be a set of cluster assignments.

	Y_1	Y_2	...	Y_p	$\sum$
Z_1	n_{11}	n_{12}	...	n_{1p}	a_1
Z_2	n_{21}	n_{22}	...	n_{2p}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Z_q	n_{q1}	n_{q2}	...	n_{qp}	a_p
$\sum$	b_1	b_2	...	b_q	

Table 3.2: Contingency table used to calculate ARI. Inspired by the table used by [10]

3.4 Chapter Summary

In section 3.1 it is specified that whole-series TSC is what will be used in this work. The different approaches, and objectives of TSC are discussed. In section 3.1.1 the different dissimilarity metrics that are to be used in the clustering models are presented, Euclidean distance and DTW.
In section 3.2 many aspects of ANN were discussed. The basic building blocks, perceptrons were presented and how they consist of weighted sums, and activation functions. Section 3.2.2 explained how ANN are trained with feed-forward computation, and SGD with back-propagation. Two special layers were presented, convolutional layers, and recurrent layers that serve their specific purpose in an ANN architecture. The section ends with discussing the two common issues of underfitting and overfitting.

Chapter 4

Review of The Literature

This chapter will contain the review of the literature.

Chapter 5

Data Exploration

In this chapter the variability, distribution and type of data used in the assignment will be explored. The exploration is divided into three sections corresponding to the three main groups of variables: The *patient meta-data*, the *input variables* and the *target variables*. The *meta-data* is the data about the patients which is not used in the classification models, but can be used to give a description of the patient demographich which makes up the dataset. The *input variables* are the variables that are inputed into the machine learning models in order to train them, and later used to make predictions about the patients' *target variables*. The target variables are then variables that the models will be trained to predict. Target variables are used both in training to correct erroneus predictions that models make during training, and to evaluate the accuracy of the model after training.

5.1 Patient Meta-data

The patient meta-data that will be considered in this section are age, gender, Body Mass Index (BMI) and blood pressure.
Figure 5.1 shows the patient distributions with regard to age, gender and BMI. As evident from the figure the patients that make up the dataset is made up of 138 males and 57 females. From the age distribution plot in figure 5.1 one can see that the majority of the patients are in the age group 60-80 years with a number of patients in the range 80-90 years. However, it should be

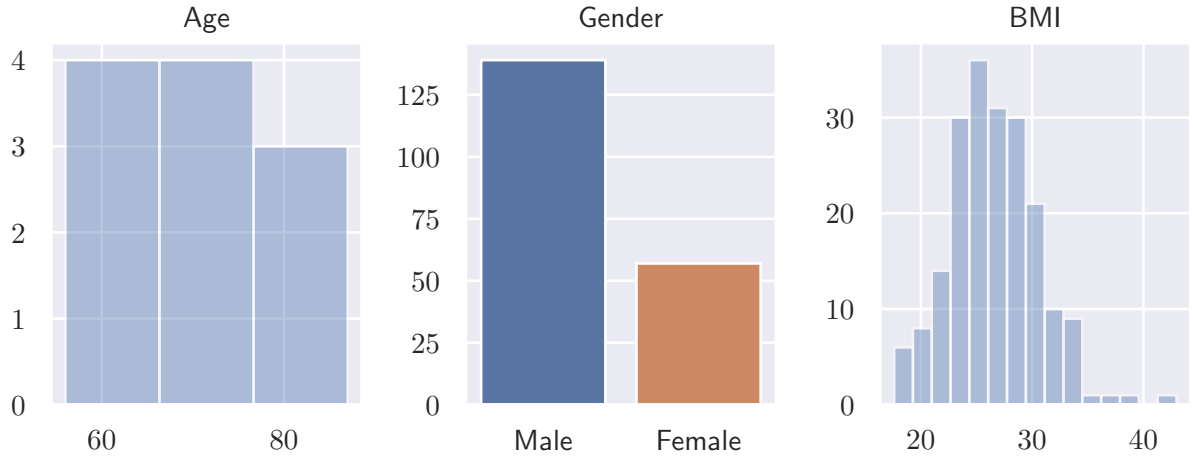


Figure 5.1: Distribution of age, gender and BMI.

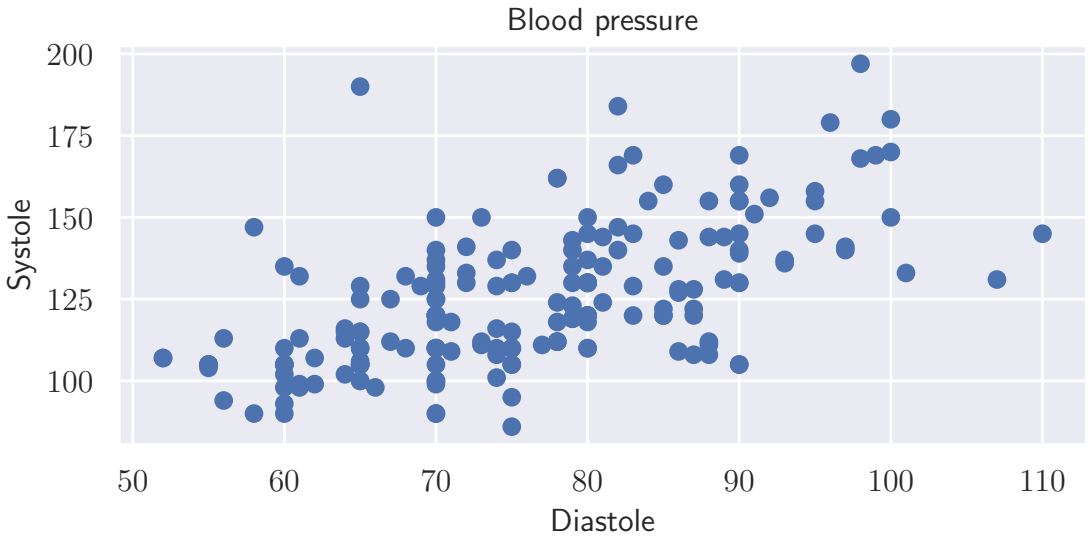


Figure 5.2: A joint distribtion plot of systolic and diastolic blood pressure of the patients.

mentioned that barely any information about the patients age has been made available. During the process of anonmization an error occured so only eleven of 200 ages were included. The BMI distribution of patients is centered around 26 kg/m^2. Even though the BMI is not always accuracte for individuals, for a population of 200 an average BMI at 26 is quite high as scores above 24.9 are considered overweight. Figure 5.2 shows the joint distribution of systolic and diastolic blood pressure among the patients.

5.2 Input variables

As mentioned earlier in section REF the different machine learning models that will be applied use two different types of input data; time-series data in the form of longitudinal strain curves, and point-values in the form of peak systolic global longitudinal strain and patient EF.

5.2.1 Peak values

As mentioned in section REFERENCE EF values below 40-50% is regarded as unhealthy with regard to probability of heart failure. Keeping that in mind, one should note that the distribution of EF values among the patients shown in figure 5.3 is centered at approximately 40% with tails going as low as 20% and as high as 70%. Figure 5.4 shows the distribution of peak systolic GLS values, four the three different views. As evident from the figure, the values are centered around -12.5 with tails going as low as -29, and as high as -2.5.

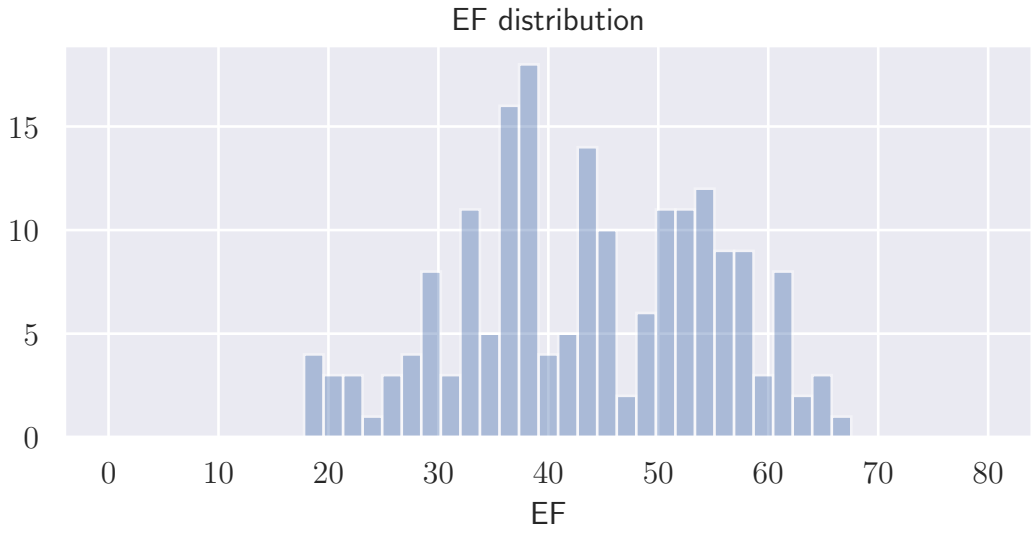


Figure 5.3: Distribution of patient EF values.

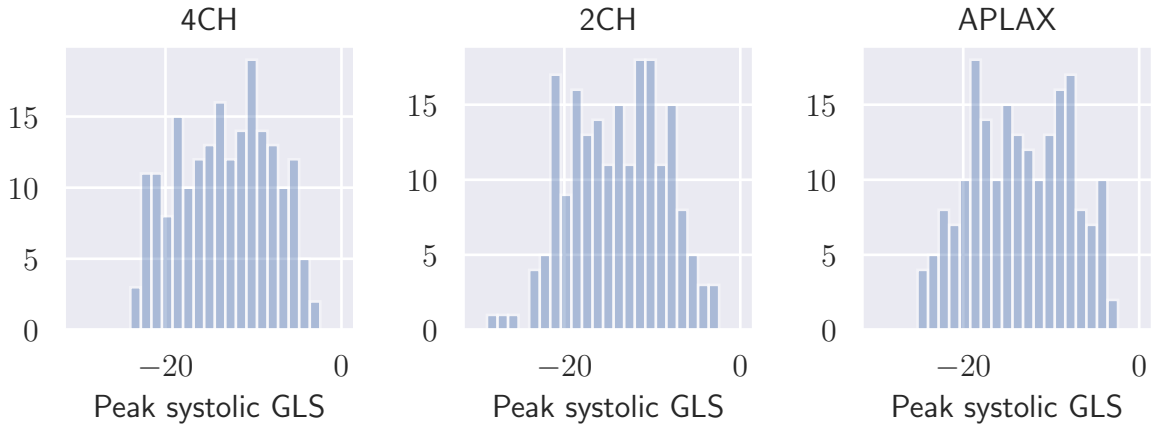


Figure 5.4: Distribution of peak systolic global longitudinal strain.

5.2.2 Strain curves

Figure 5.5 shows what a typical set of strain curves look like for a patient. Only the six regional strain curves, and the one global strain curve from the 4CH view have been included as they are fairly similar across the different views. Since the data from the different patients have been taken at different times, and possibly with different ultrasound machines factors such as number of samples per strain curve, and the frame rate of the particular ultrasound machine during an examination. Each strain curve has a standardized length of one heart cycle, due to this different curves have different number of samples. Figure 5.6 shows the distribution of frame rates, and number of samples among the total number of strain curves.

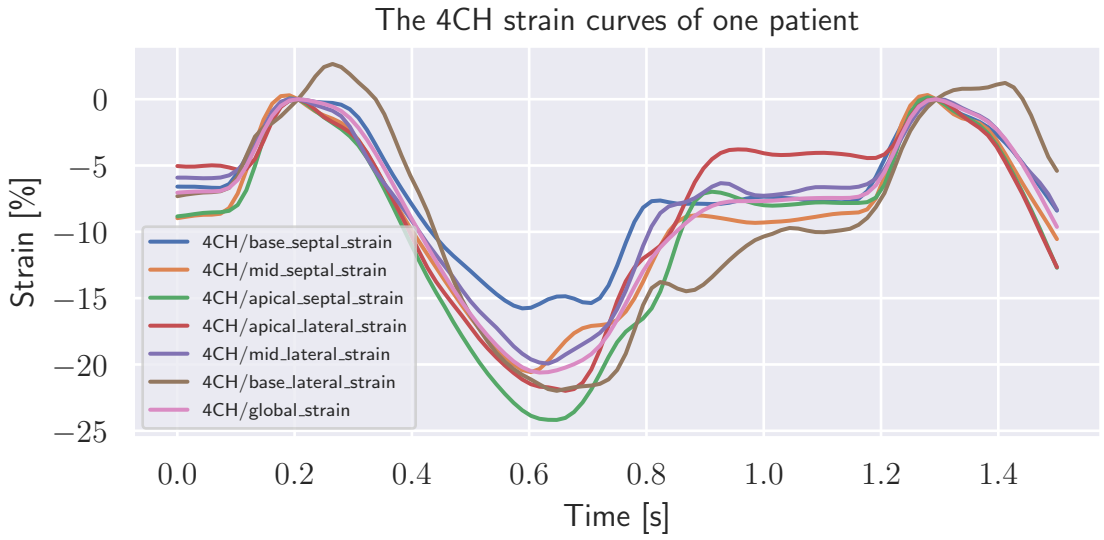


Figure 5.5: Plot of the global and regional longitudinal strain curves of one patient in the 4CH view.

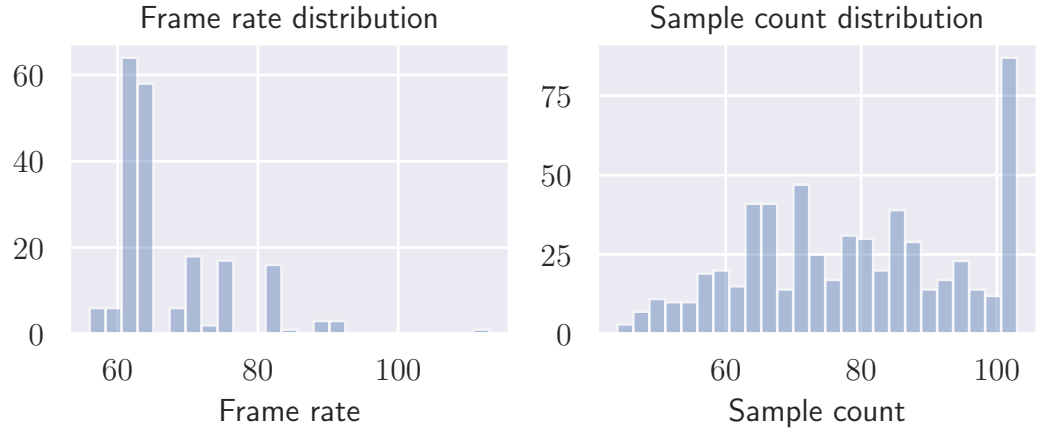


Figure 5.6: Distribution of the frame rate used in the ultrasound imaging used to obtain the strain curves (left), and sample count of the different strain curves (right).

5.3 Target variables

Figure 5.7 shows the distribution of heart failure among patients (left), and the distribution of different diagnoses (right). Since the dataset has approximately as many patients with a heart failure diagnosis as without, it can be considered balanced in that regard. With regard to the different patient diagnosises, their rate of occurance is not uniform in this dataset. The control group of healthy individuals consists of 30 patients. The groups of patients with STEMI, and NSTEMI diagnoses consist of 60 and 39 patients respectively. Finally, the group of patients with heart failure, but with a non-stemic indication (labelled OTHER in left barplot in figure 5.7) consists of 70 patients. To simplify the classification problem this work will only attempt to separate healthy patients from unhealthy patients. All the 169 diagnosed patients are therefore grouped together under the label *unhealthy*.

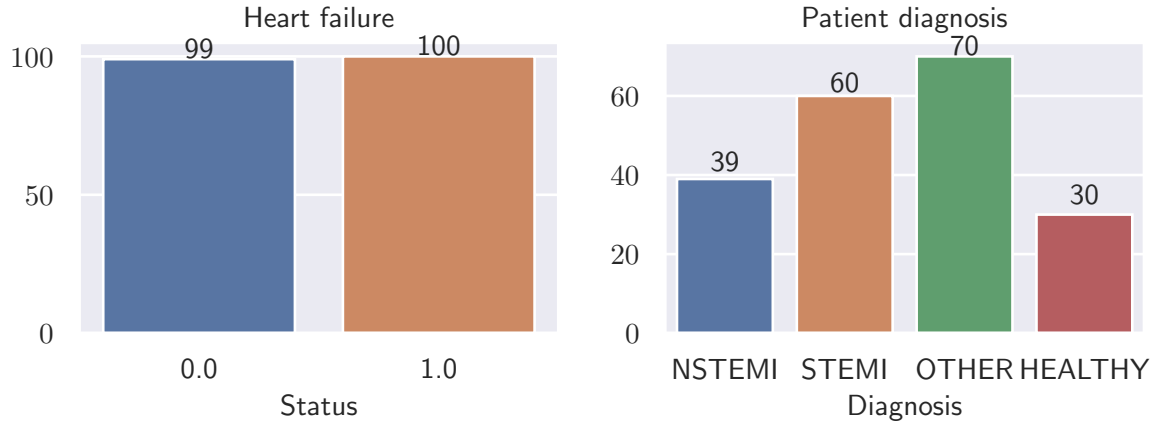


Figure 5.7: The distribution of heart failure and different diagnoses within patients.

To illustrate the diagnostic power of peak systolic strain, and EF 5.8 shows the distribution of EF for patients with and without heart failure (left), and the distribution of EF for patients with and without a heart disease diagnosis (right). Figure 5.9 shows the distribution of peak systolic GLS values for patients with and without heart failure, and figure 5.10 shows the distribution of peak systolic GLS values for patients with and without a heart disease diagnosis. From the samples used to produce the left plot in figure 5.8 and figure 5.9 it seems as though the heart failure patients are more separable with the EF values than with the GLS values. With regard to separability of patients with diagnoses and patients in the control group it seems as though the right plot in figure 5.8, and figure 5.10 follow the same distribution as the heart failure patients. However, it is hard to make an evaluation on this since the sample size of the control group is much smaller than the group of patients with a heart disease diagnosis.

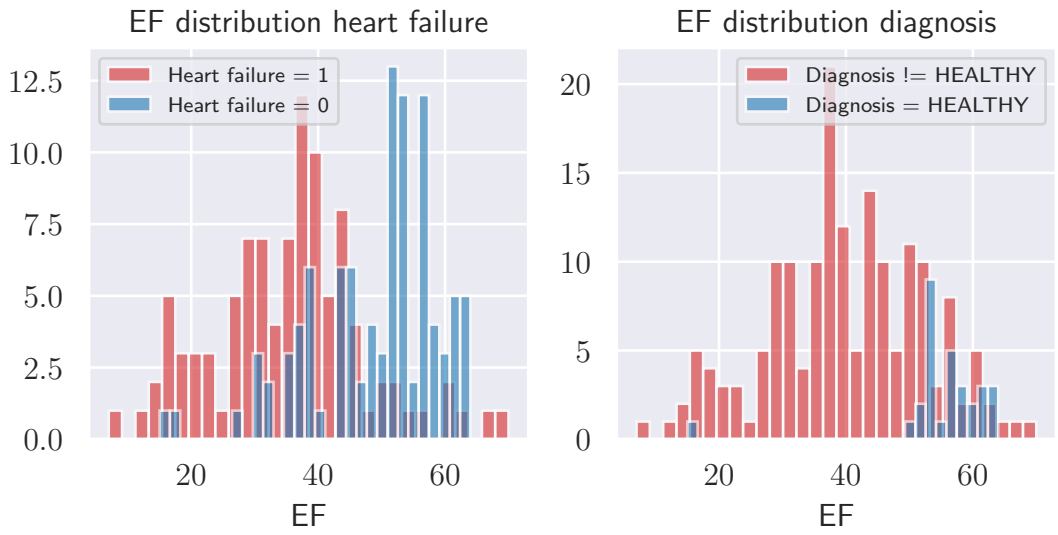


Figure 5.8: Distribution of EF for patients with and without heart failure (left), and distribution of EF for patients in the control group, and patients with a diagnosis.

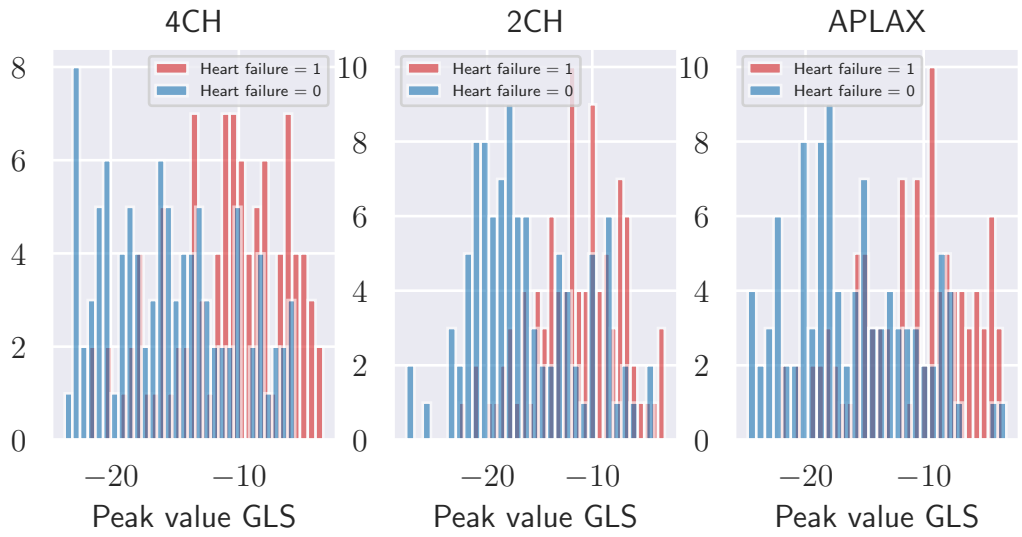


Figure 5.9: Distribution of GLS for patients with and without heart failure.

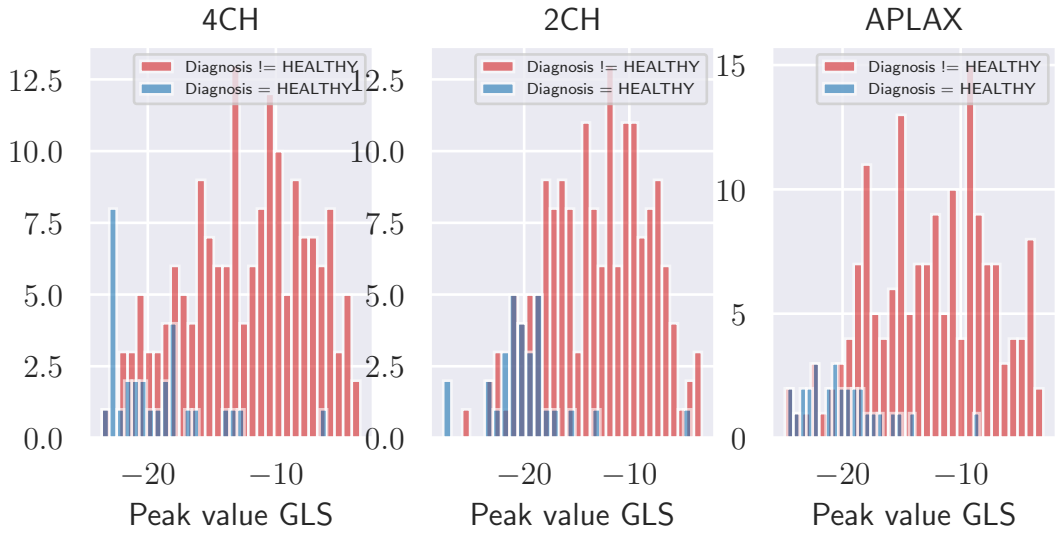


Figure 5.10: Distribution of GLS for patients in the healthy control group, and the other patients.

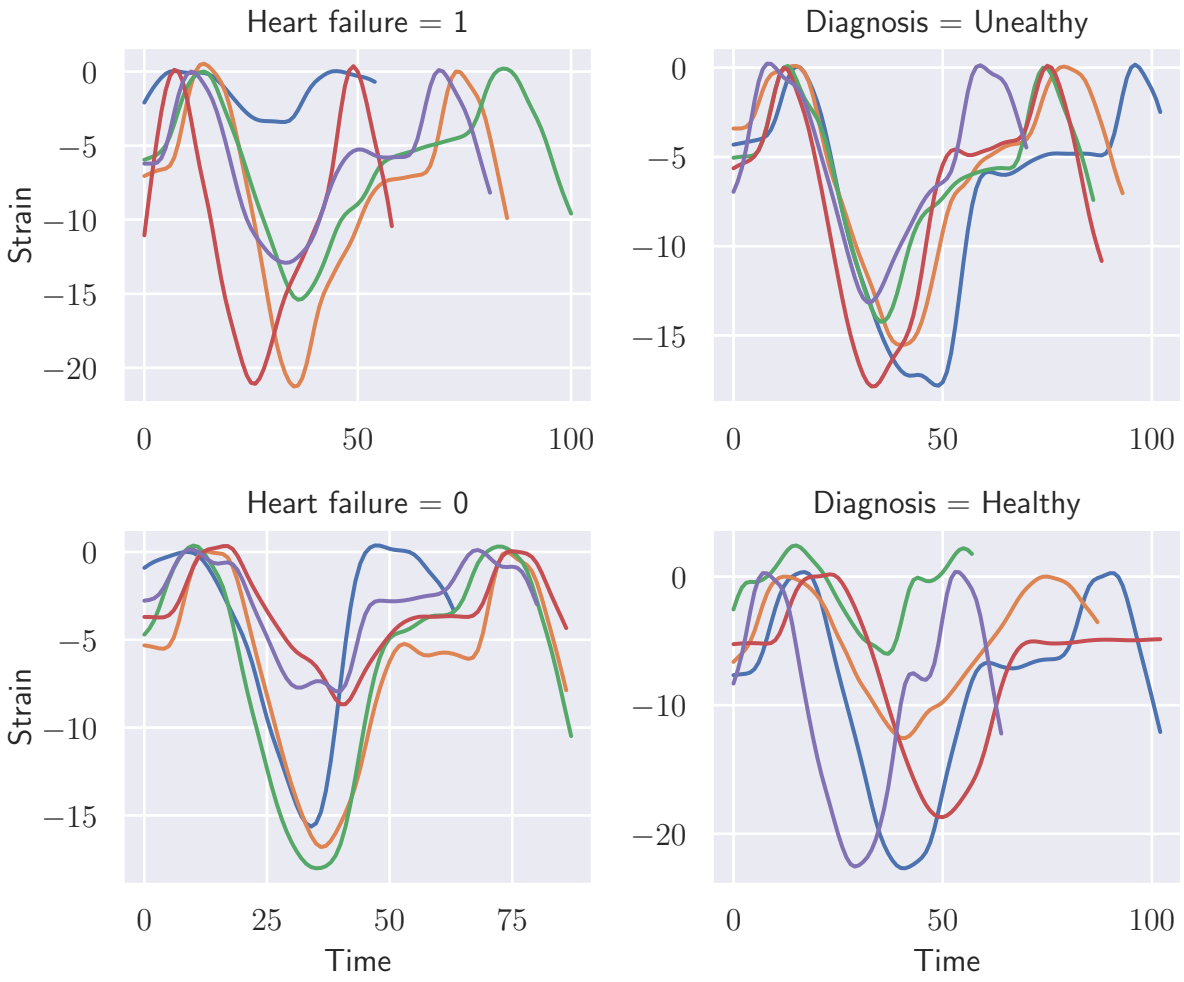


Figure 5.11: The left column shows five sample GLS curves for patients with (top), and without (bottom) heart failure. The right column shows five sample GLS curves for unhealthy (top) and healthy (bottom) patients.

Figure 5.11 shows five random sample GLS curves from all views for patients with different conditions. GLS curves for patients with, and without heart failure is illustrated on the column to the left, and patients with and without a heart disease diagnosis is illustrated to the right. For the curves it is not easy to visually discern the difference between heart failure patients, and diseased patients based on the shape.

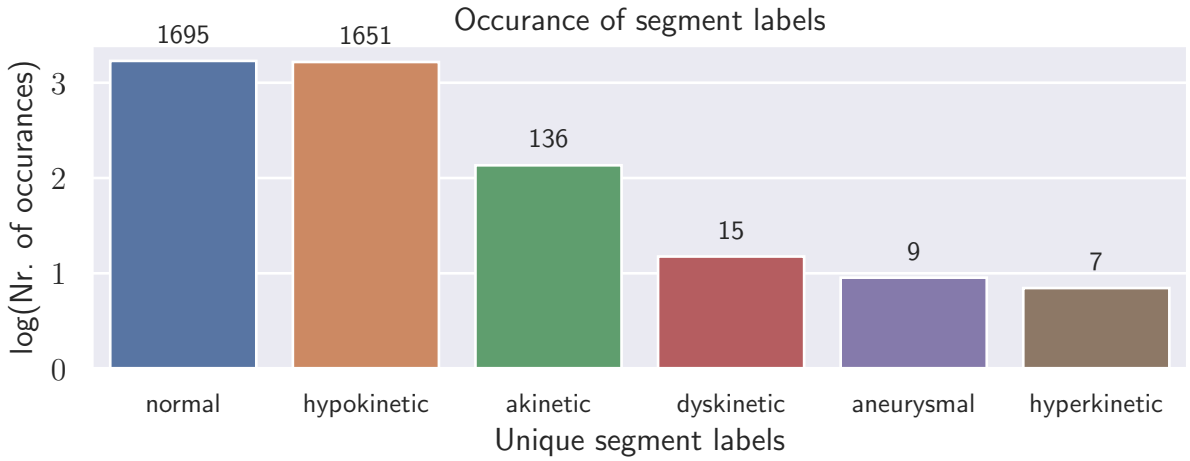


Figure 5.12: Distribution segment indication labels.

Figure 5.12 shows the distribution of the different segment indications, for all the left ventricle segments of all the patients in the dataset. Since the occurance of indications other that "normal" and "hypokinetic" are very rare, the occurance axis has been represented logarithmicly. The imbalance of segment-indication labels illustrated in figure 5.12 means that it will challenging for any statistical model to perform well in the classes with low occurance. To counteract this the taxonomy of the labels is changed such that the classification problem becomes binary with the labels *Normal* and *Not normal*, similarily as was done with the patient diagnoses. The dataset is then fairly evenly distributed with 1695 *Normal* labels and 1818 *Not normal* labels. Figure 5.13 shows five random sample RLS curves that represent the different segment indication labels. Figure 5.13 shows five random sample RLS curves that represent the different labels. In this case, it is easier to see the difference between the different segmental labels. For the RLS curves that are labelled as hyperkinetic, one can see that compared to the curves regarded as normal these curves in general have troughs in strain that go further down than the normal curves. The RLS curves regarded as normal rarely go below -20, whereas the hyperkinetic curves regularily pass -20 and some of them go as low as -30. This observation is consistent with the label "hyperkinetic" indicating that the segment has a larger range of motion than normal. The curves with the hypokinetic, akinetic and dyskinetic all show similar characteristics of various degrees, the curves within these three categories have peaks, and troughs that are smaller in magnitude than the curves that are considered normal. The RLS curves regarded as akinetic, and dyskinetic are also smaller than the curves with the hypokinetic label. These observations also agree with the label names, as segments labelled hypokinetic, dyskinetic and akinetic all indicate that the segments have different degrees of reduced range of motion. The RLS curves that are labelled aneurysmal have significantly more positive strain than the curves with any other label. Two curves have peaks as high as 20, whereas the curves with the other labels rarely pass 5.

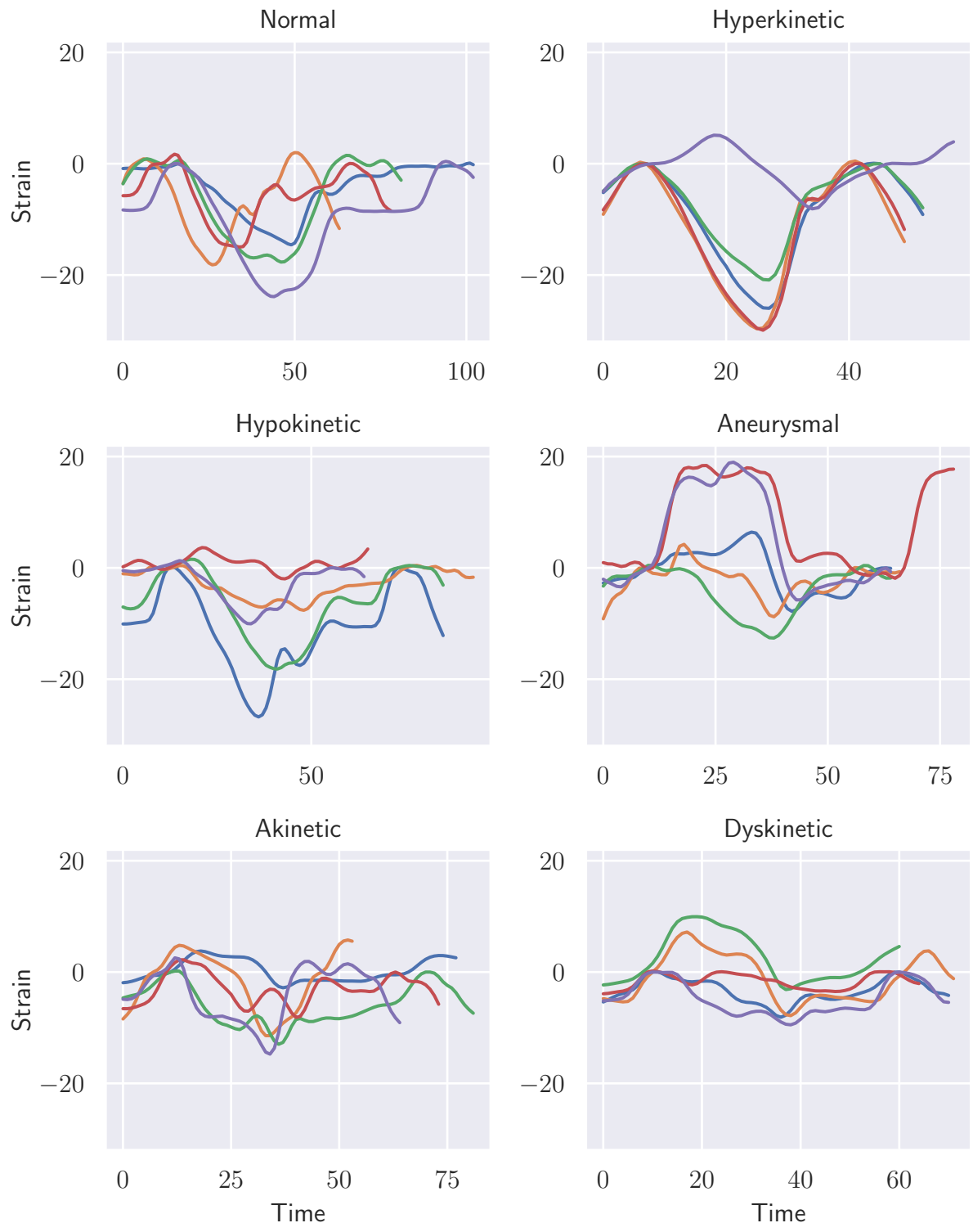


Figure 5.13: Each plot in this figure shows five random sample RLS curves that are labelled with the indication in the title of the plot.

Chapter 6

Method

6.1 Description of The Datasets

Since the different ML models require different types of input data the, datasets have been divided into two main categories: The peak-value datasets and the time-series datasets.

6.1.1 Time-series Datasets

Nr	Input variables	Shape
1	Single RLS curves	$(3600, 1)$
2	RLS curves	$(200, 18)$
3	GLS curves	$(200, 3)$
4	Strain curves	$(200, 21)$

Table 6.1: Time-series datasets. The "Shape" parameter is indicates: (Number of objects in the dataset, Number of curves used to represent each individual object). The curve length is not included in the shape parameter because it differs for different curves.

Table 6.1 shows the different time-series datasets that will be used. All the datasets except *Single RLS curves* will be used to predict whether or not the patient is diagnosed, and whether the patient has heart failure. Recall that the different diagnosises are described in section REFERENCE, and there occurance rate are illustrated in figure 5.7.*Single RLS curves* will be used to predict the segment indications shown in figure 5.12 and described in section REFERENCE. The point of classifying individual segments of a patients left ventricle is that if a single segment is found to be *not normal*, this would also mean that the patient can be considered as *not healthy*. As mentioned in the description of table 6.1 the "Shape" parameter shows how many objects each dataset has, and how many curves are associated to each object. Since each ultrasound examination takes ultrasound inspections from three views (four chamber, two chamber, and APLAX chamber), each patient has three views to estimate a GLS curve from. Since each GLS curve, also can be divided into six RLS curves, there is a total of 21 strain curves per patient. Since each patient has 18 RLS curves, there are $18 \times 200 = 3600$ curves that make up dataset number 1. For datasets two to three it will also be experimented with wether using data from a single view performs better than data from all views. For dataset two that means that the number of curves used to represent an object will be either 6 or 18, for dataset three

it will be either 1 or 3 curves and for dataset four patients will be represented with either 7 or 21 curves. Both the ANN, and the TSC model are applied on the datasets listed in table 6.1.

6.1.2 Peak-value Datasets

Nr	Input variables	Shape
1	Peak systolic RLS values	$(200, 18)$
2	Peak systolic GLS values	$(200, 3)$
3	Peak systolic strain values	$(200, 21)$
4	Peak systolic RLS, and EF values	$(200, 19)$
5	Peak systolic GLS, and EF values	$(200, 4)$
6	Peak systolic strain, and EF values	$(200, 22)$

Table 6.2: Peak-value datasets. The "Shape" parameter is indicates: (Number of objects in the dataset, Number of dimensions used to represent each individual object).

Table 6.2 shows the different peak-value datasets. All the datasets will be used to predict the diagnosis of patients, and whether the patient has heart failure. Single peak systolic RLS values were not considered suited for PVC or PVSC models to predict segment indication, because the best model one can hope for from a one dimensional point-value dataset is a form of threshold classifier. The reason that there are more peak-value datasets than there are time-series datasets, is that the peak-value version of three datasets in table 6.1 have been combined with EF to determine whether a combination of peak systolic strain, and EF can have a higher predictive power than strain alone.

6.2 Clustering

The implementations of the two clustering models that are applied in this work are described together in the same section because conceptually, they are almost identical. It is only the method used to measure dissimilarity that separates the PVC and TSC models. The general implementation of the clustering models is illustrated in figure 6.1. Time-series datasets are preprocessed before dissimilarity measurement, peak-value datasets are not. In the coming subsections the processes in each of the boxes in the flow diagram will be expanded upon.

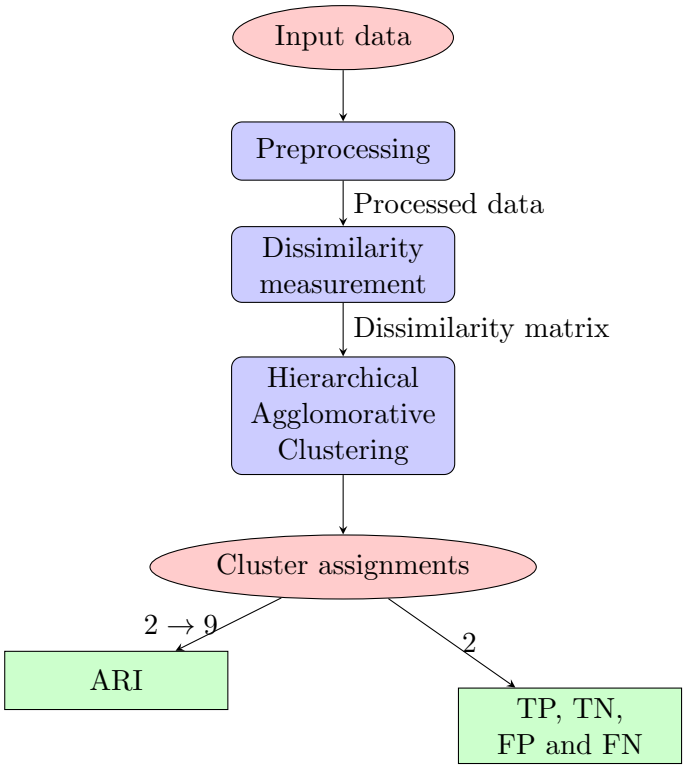


Figure 6.1: A flow diagram to give an overview of how the PVC and TSC models are implemented and evaluated.

6.2.1 Time-series Preprocessing

Preprocessing of the time series is done because it is known that the DTW distance is sensitive to absolute difference, and offsets of time series. In addition to clustering the longitudinal strain time series without preprocessing three forms of preprocessing were tested to see whether they could improve the predictive performance of the clustering algorithm: Normalization, scaling and Z-score normalization. The normalized version of a time series ($\{x_t\}_N$) is calculated by equation (6.1). The smallest recorded value in the time series ($\min\{x_t\}$) is subtracted from the time series ($\{x_t\}$), then the time series is divided by the difference between the highest recorded value ($\max\{x_t\}$), and lowest recorded value in the time series.

$$\{x_t\}_N = \frac{\{x_t\} - \min\{x_t\}}{\max\{x_t\} - \min\{x_t\}} \tag{6.1}$$

The "scaled" version of a times is calculated in a similar fashion. Scaling can be considered as normalizing a time series with regard to the highest, and lowest recorded values of the entire set of time series it is being compared to. If one lets $\{\{x_t\}\}$ represent the set of time series to be scaled, $\min\{\{x_t\}\}$ represent the smallest recorded value in the entire set of time series and $\max\{\{x_t\}\}$ represent the highest recorded value in the set of time series, the scaled version of a time series ($\{x_t\}_S$) is given by eqation (6.2).

$$\{x_t\}_S = \frac{\{x_t\} - \min\{\{x_t\}\}}{\max\{\{x_t\}\} - \min\{\{x_t\}\}} \tag{6.2}$$

The Z-score normalization is done by transforming each observation of a time series to it's Z-score. The Z-score of an individual time-series observation is calculated by subracting the expected value of the time series, and dividing by the standard deviation. The unbiased estimators used to calculate the expected value, and standard deviation of a time series are given in equations (6.3), and (6.4) respectively. The Z-score normalized version of a time series ($\{x_t\}_Z$) is calculated using equation (6.5)

$$\hat{\mu} = \frac{1}{n}\sum_{t=1}^{n} x_t \tag{6.3}$$

$$\hat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{t=1}^{n}(x_t - \hat{\mu})^2} \tag{6.4}$$

$$\{x_t\}_Z = \frac{\{x_t\} - \hat{\mu}}{\hat{\sigma}} \tag{6.5}$$

Figure 6.2 illustrates how the different preprocessing methods work on the 4CH GLS curves of four random patients. By comparing 6.2a and 6.2d one can see that scaling preserves both the relative offsets and relative size differences between the curves. From 6.2b one can see that though normalization preserves the offsets of the curves, the relative sizes are not. From 6.2c one can see that Z-score normalization preserves the offsets of the curves, the relative sizes are only preserved to a certain extent. In addition, the normalized, and scaled curves are constricted between 0 and 1, while the Z-score normalized curves are not.

6.2.2 Dissimilarity measurement

When estimating dissimilarity between patients represented by a peak-value dataset Euclidean distance was used. To measure the dissimilarity between longitudinal strain curves in the TSC model DTW distanse was used. Recall that the DTW distance between to time series is the length of the shortest DTW path between them. To calculate the DTW distance the **dtaidistance 1.2.5** library was used. The **dtaidistance** library is used by the DTAI Research

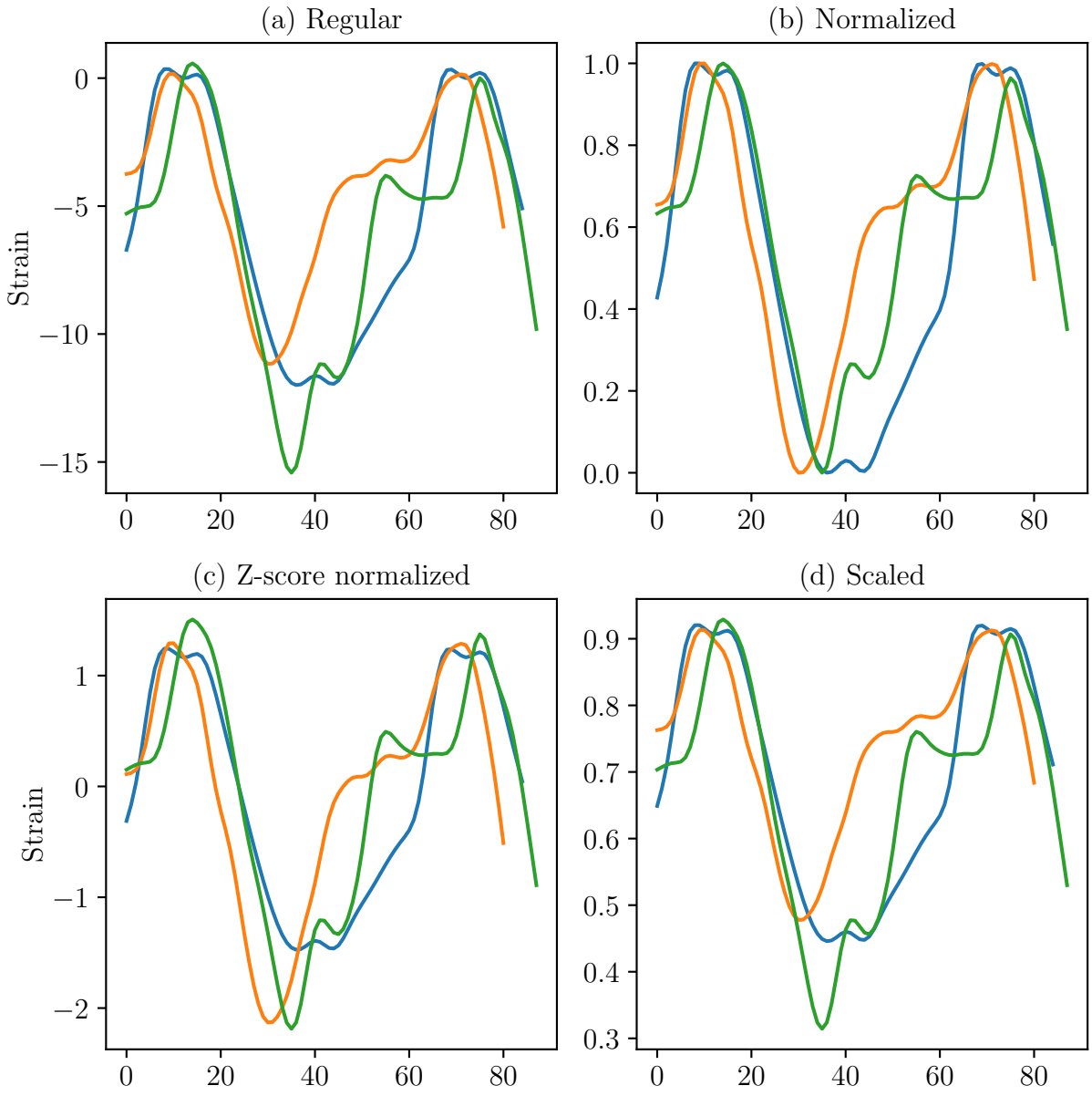


Figure 6.2: Four plots of three random 4CH GLS curves that are preprocessed in the three different ways. (a) no preprocessing, (b) normalization, (c) Z-score normalization and (d) scaling

Group to measure distances between time series. To encapsulate all the dissimilarity between patients in a single matrix, one first has to calculate one matrix of DTW distances for each of the time series used to represent patients. Say that a patient was represented using the GLS curves in the three views. To calculate the dissimilarity matrix one would first estimate the DTW distance between all the 4CH GLS curves, then all the 2CH GLS curves and finally all the APLAX GLS curves. By adding the three matrices of DTW distances together one gets the dissimilarity matrix.

6.2.3 Hierarchical Agglomorative Clustering

As mentioned in section REFERENCE, the hierarchical agglomorative clustering algorithm takes inn the dissimilarity matrix, and starts with every patient being represented by one cluster each. For each number of possible clusters then two clusters are merged based on minimizing one of the six linkage criterions. There are also various options of distance metrics that can be used by the clustering algorithm to measure difference between elements of the dissimilarity matrix, but in this work only Euclidean distance is used. The clustering algorithm used for time-series data is implemented using the **scipy.cluster.hierarchy 1.4.1** library, and in the TSC model all six linkages detailed in section REFERENCE are tested. In the PVC models a more holistic implementation is applied using the **scikit-learn 0.22.1** library. The implementation of the PVC models does both the dissimilarity and clustering using one library, and because the scikit-learn library supports fewer linkage criteria only the single, complete, average and ward linkages are tested.

6.2.4 Cluster Assignment Evaluation

When evaluating a specific TSC, or PVC clustering model, the model is evaluated at two to nine cluster cluster centers. For the cluster assignments given by evaluating the model at two cluster centers the models TP, TN, FP and FN are calculated. These metrics are then used to estimate the models accuracy, sensitivity, specificity and DOR. The cluster assignments for a clustering model evaluated at two cluster centers can be either 1 or 2, and since clustering is a form of unsupervised machine learning it is not given whether cluster 1 or 2 corresponds to the 1 or 0 of the target variable. Therefore, the evaluation metrics are calculated twice for each clustering model evaluated at two cluster centers, once where cluster 1 corresponds to target variable 1, and once where cluster 2 corresponds to target variable 1. The calculation that yields the highest accuracy is kept, and the other is disregarded. In addition to these matrics the ARI is used to evaluate all the cluster assignments yielded from evaluating a clustering model at between two to nine cluster centers. The ARI is used because it can give a measure of how correlated the distributions of the cluster centers are with regard to the distribution of the target variables. This can give insight into whether a clustering model evaluated at a higher number of cluster centers than two is better at capturing a particular target variable. In the heart failure, and patient diagnosis case studies there are twelve different datasets, four types of preprocessing, and seven different linkages tested. This yields a total of 336 variations of the TSC model that are tested in the heart failure, and patient diagnosis case studies. In the segment indication case study, there are only 28 variations of the TSC model tested since there is only one dataset. For the PVC models there are six datasets tested, and four linkages tested yielding 24 variations of the PVC model tested in the heart failure and patient diagnosis case studies.

Strain curves used	Views used	Nr. of time series	Nr. of trainable parameters
GLS, or single RLS curves	Single view	1	39,457
GLS curves	All views	3	43,553
RLS curves	Single view	6	49,697
RLS curves	All views	18	74,273
GLS and RLS curves	Single view	7	51,745
GLS and RLS curves	All views	21	80,417

Table 6.3: This table shows the total number of trainable parameters of the ANN, for different number of time-series inputs.

6.3 Artificial Neural Network

6.3.1 Preprocessing

Two methods of preprocessing were tested on the data used as input for the ANN in addition to testing the ANN models witout preprocessing. Since neural networks with recurrent layers perform better when the sample rates of the input time series are equal, as they usually aren't correlated with the target variable. Since the frame rate of the ultrasound videos vary from patient to patient, the frame rates of the longitudinal strain curve time series also vary, and in this case sample rate is not correlated with heart failure, patient diagnosis or segment indication. To counteract with this it was tested whether upsampling all the strain curves to the highest sample rate, or downsampling them all to the lowest frame rate would affect the performance of the ANN.

6.3.2 Architecture

The architecture of the ANN used in this work was not designed by the author himself, as there sadly was not enough time. The architecture was designed by student Benjamin Nedregård and was used to estimate heart phase, and patient health using blood flow curves as input. The reason why this architecture was applied is because it showed great promise when applied to blood flow time series, which share charactaristics with left ventricle longitudinal strain time series. The network was implemented using the **keras** with **tensorflow 2.1.0** as backend. The architecture used for the ANN is illustrated in figure 6.3. One aspect that is not shown in figure 6.3 is the total numbe of trainable parameters of the architecture. The reason for this is because it varies based on the shape of the dataset it is applied on. In section 6.1.1 the different time-series datasets that are used in this thesis are detailed. Recall that the different datasets will use different combinations of GLS and RLS curves from one or all of the three ultrasound views. Because of this the number of curves used as input for the ANN can be 1, 3, 6, 7, 18 or 21.
Since the author did not design the architecture of the network, a thorough defence of the architecture will not be given. However, a brief explanation of the properties the different layers contribute with to the model as a whole will be given. The two first layers in the ANN are convolutional, and are intended to detect simple structures in the time series such as linear regions, curved regions and rapid changes in the signal. The recurrent layer is intended to detect time dependant relations of the signal such as periodicity and frequency. Regularization terms are added to the outputs of the convolutional, and recurrent layers to attempt to bias the weights toward zero, which is a technique used to avoid overfitting.Finally the dense layers are intended to connect the features extracted by the previous layers to specific values the target

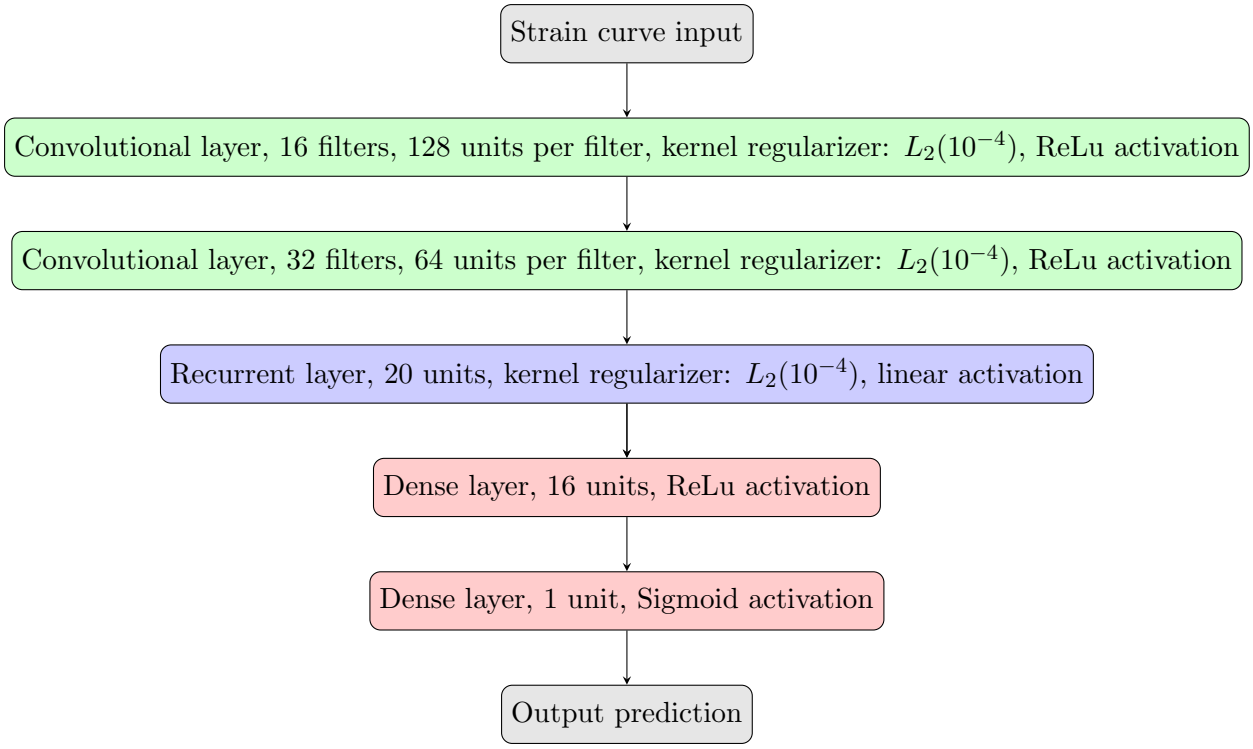


Figure 6.3: A block diagram illustrating the architecture of the ANN used in this work.

variable can have, and make a prediction.

6.3.3 Training and Validation

Binary cross entropy was used as loss function during training of the variations of the ANN model. Each variation was trained for five epochs, using back propagation and SGD. The ADAM learning rate optimizer was used with an initial learning rate of 10^{-3} with the intention of avoiding that the loss function of the ANN got stuck in local minima during training. The baises of each layer were initialized as zeros, and the weights of the individual units were initialized by sampling from the standard normal distribution function. .

To validate the ANN models, 10-fold cross-validation was used. N-fold cross validation of a model-dataset combination entails dividing the dataset into N chunks of equal size, and preferably with an approximately equal distribution of target-variable values in each chunk. Then in N rounds, called *folds* $N-1$ chunks are used to train the model and the final chunk is used to test the model. For each round one also changes which chunk is used to test the model such that it is able to attempt making a prediction on every value of the dataset. When validating the variations of the ANN using cross-validation the number of TP, TN, FP and FN attained during each fold were recorded and added together after all the folds were complete. The sum of all TP, TN, FP and FN attained during cross-validation are used to estimate the models' accuracy, sensitivity, specificity and DOR. Since there are a total of twelve datasets and three types of preprocessing tested, there are a total of 36 variations of the ANN model applied in the heart failure and patient diagnosis case studies. In the segment indication case study there is only one dataset, and three forms of preprocessing tested, so there are only three variations of the ANN model tested.

6.4 Peak-value Supervised Classifiers

Eleven base supervised classifiers are included in the PVSC model group, they are all implemented using the **scikit-learn 0.23.1** library. In this section a short description of the theory behind these classifiers, and the hyperparameters used in this work will be given. The PVSC models are validated using 10-fold cross-validation in the same manner as the ANN models.

6.4.1 Multi-layer Perceptron

The MLP is mentioned earlier described earlier in section 3.2.1. This MLP is configured with a single dense layer with a 100 neurons with the ReLu activation, and an output layer of a single neuron since the classification problem is binary. It is trained with using SGD with back-propagation, with ADAM as gradient descent optimizer, and an initial learning rate of 10^{-3}.

6.4.2 K Nearest Neighbors

K Nearest Neighbors (KNN) is a machine learning model that can be used for classification and for regression. KNN are described as a form of *lazy learner* because it does not extract generalized rules from the training set that are used to relate the input, and target variables, but instead memorizes the dataset [11]. When used for classification the target variable is predicted based on the objects from the training set which are "nearest" in terms of input variable values. Hence, there are two central features that define a KNN classifier: The number of neighbors used for comparison, and the distance metric used to measure proximity to its neighbors [12]. In the implementation used in this work, the model was constricted to only consider five closest neighbors weighted equally, and use Euclidan distance as a distance metric. The implementation uses a combination of three algorithms to compute the nearest neighbors: BallTree, KDTree and brute force search. The BallTree and KDTree algorithms are constricted to a maximum of 30 leaves.

6.4.3 Support Vector Classifier

Support vector machines were originally implemented as a type of binary classifier that could classify linearly separable variables, as classifiers they are referred to as Support Vector Classifier (SVC). Under ideal conditions SVC transform input variables to a set of hyperplanes where the target variable values are linearly separable [13]. The transformation used depends on what kernel is used, some examples of kernel functions include: Linear kernal, Radial Basis Function (RBF) and sigmoid function. In this work two versions of the SVC are tested, one with a linear kernel and one where the RBF is used. The RBF is given in in equation (6.6), where γ is equal to 2. Both SVC applied use an L_2 regularization penalty to avoid overfitting. For the RBF SVC parameter C which is the inverse strength of the regularization is set to 1. For the linear SVC C is set to 0.025.

$$\mathrm{RBF}(\mathbf{x_1}, \mathbf{x_2}) = e^{\gamma(\mathbf{x_1} - \mathbf{x_2})^2} \tag{6.6}$$

6.4.4 Gaussian Process Classifier

Gaussian Process (GP) are a probabilistic machine learning technique that can be used for regression, and classification tasks. Similar to SVC they perform best when the relationship between the input variables and the target variable are linear, but by the use of what is called *basis functions* they can map the input variables to a hyperplane where the targets are linearly seperable [14]. One can say that basis functions are for GP, what kernels are for SVC. The defining difference are that GP are probabilistic while SVC are deterministic. Where SVC work

with a single kernel GP work with an infinite set of basis functions, and much of the training process amounts to finding an optimal linear combination of the set of kernals available [14]. The implementation of the GP classifier in this thesis uses an RBF function as covariance function with γ equal to 0.5. The implementation uses the "L-BFGS-B" algorithm to optimize the basis functions parameters during training, and sets a maximum of 100 iterations of Newtons method during predict operations.

6.4.5 Naive Bayes Classifier

The naive bayesian classifier used in this work is specifically a gaussian naive bayesian classifer, it is a probabilistic classifier based on Bayes rule, and the assumption that individual input features are independent. If one lets $\mathbf{X}$ be the training input data, $\mathbf{t}$ be the training target data, x_{new} be a piece of new input data and t_{new} be the corresponding new target. Bayes rule in this context is then given by equation (6.7).

$$\mathrm{P}\left(t_{new} = k | \mathbf{X}, \mathbf{t}, x_{new}\right) = \frac{\mathrm{P}\left(x_{new} | t_{new} = k, \mathbf{X}, \mathbf{t}\right) \mathrm{P}\left(t_{new} = k\right)}{\sum_j \mathrm{P}\left(x_{new} | t_{new} = j, \mathbf{X}, \mathbf{t}\right) \mathrm{P}\left(t_{new} = j\right)} \tag{6.7}$$

What the naive bayesian classifier asssumes that the likelihoods of the input features follow gaussian distributions where the mean and variance are found using maximum likelihood estimation. Predictions are then made by yielding the label with the distribution from which the new data object is most likely to have been sampled from.

6.4.6 Quadratic Discriminant Analysis

Discriminant analysis classifiers are classifiers that are able to set polynomial thresholds in the input feature space. Linear discriminant analysis classifers are able to set multiple linear thresholds, and as the name implies quadratic discriminant analysis classifiers are able to set quadratic boundaries [15].

6.4.7 Decision Tree Classifiers

Decision tree classifiers are classifiers that create hierarchies of rules that are used to make predictions of the target variable based on the values of the input variables [11]. The advantages of decision tree classifiers is that they are highly interpretable because of their rule based structure, and do not require as much data as many other classifiers. The main disadvantage of decision trees are that they are prown to overfitting unless restrictions are set as to how many branches can be grown, and what the maximum depth of the tree can be. [11]. For the implementation of the single decision tree classifier used in this work the "Gini impurity criterion" is used to choose which of the existing nodes would yield the split of highest quality, it uses the "best" strategy to choose splits at a node and is allowed a max depth of five nodes.

"Extra Trees", and "Random Forest" are two ensamble methods that are based on initiating many decision tree classifiers. The Random Forest method makes a prediction by averaging predictions of many different Decision Tree classifiers that are trained on separate random partitions of the dataset [16]. When choosing which node to split, a randomized subset of the input features are used to make the choice. These two additions of random behaviour are meant to decouple the prediction error of individual trees, such that an averaged prediction of all the trees will have higher accuracy than any single tree, and will reduce the probability of overfitting [15]. The implementation of the Random Forest classifer used in this work instantiates ten different decision trees that use the "Gini impurity criterion" to measure the quality of a split, the trees are allowed a maximum depth of five nodes and are only allowed to consider a single feature when finding the best split.

The Extra Trees classifier, also known as "extremely randomized trees" works similarily to Random Forest, but introduces one more source of "randomness" to the mix [15]. When training a Random Forest classifier one attempts to estimate the threshold of a node-split such that the discrimination between data objects is maximized based on their target variable value, for an Extra Trees classifier multiple thresholds are picked at random, and the threshold than maximizes discrimination of data objects based on their target variable value is chosen [15]. The implementation of the Extra Trees classifier used in this work uses 100 different decision trees, each using the "Gini impurity criterion" to measure the quality of a split and there is set no limitation to how deep the trees can be, or the number of features that can be considered during a node-split.

6.4.8 Ada Boost Classifier

The Ada Boost classifer is an ensamble models like Random Forest, and Extra Trees. What makes Ada Boost different from Random Forest, and Extra Trees is that it is not neccesarily restricted to using Decision Tree classifiers as base classifiers, and it uses a voting system to make predictions and train the base classifiers. The training process works by assigning weights to each of the training objects. In the first round of training, each object is assigned the same weight $1/N$, where N is the number of training objects, in the preceding rounds the weights are updated by assigning smaller weights to the objects that the model is able to predict correctly and bigger weights to the objects predicted wrong, the training algorithm is repeated until the maximum number of iterations is reached or the accuracy converges to a fixed value [15]. For the implementation of the Ada Boost classifer used in this work use a Decision Tree classifier with the same parameters as the classifier in section 6.4.7 is used, only with a maximum depth of one.

6.5 Presentation of Results

In chapter 7 the results of the different models will be presented in the form of three case studies. Each case study will focus on a single target variable, and aims to find which model group performs best at predicting the target variable in question. Recall that the three target variables that will be considered in this thesis are: Heart failure, patient diagnosis, and the indication of individual left ventricle segments. As mentioned earlier in the chapter, four models will be tested. The case studies will first deal with each model individually, where variants of the models with different hypermarameters will be tested on the different datasets. Then, the best performing model variation within the four main models will be used to for comparison. The supervised models will be assessed with the metrics: accuracy, sensitivity, specificity and DOR. The clustering methods evaluated at two cluster centers will be assessed with the same methods as the supervised models. The clustering methods evaluated at two to nine cluster centers will also be assessed with ARI to determine whether the models evaluated at a higher number of cluster centers could fit the data better.

The results of each model group in every case study will be presented in the form of a distribution plot of the DOR, a scatter plot of sensitivity versus specificity, a table showing the five model variations that attain the highest DOR, and if the model group is a clustering model a table of the the five model variations that attain the highest ARI will also be presented. Recall the definition of the DOR from equation (3.6), if a DOR is between 0 and 1 it indicates that the product of FP, and FN is greater than the product of TP, and TN meaning that the performance of the classifier model is bad. If the DOR attained by a model is greater than 1, that at least means that it attained more correct predictions than wrong predictions. For a balanced dataset random guessing should attain accuracy, sensitivity and specificity scores of approximately 50%, hence scores below 50% can be considered as bad. For unbalanced datasets

such as the patient-diagnosis dataset where there are 170 positives and 30 negatives it becomes a bit more complicated. Only guessing 1 will yield an accuracy of 85%, a sensitivity of 1 and a specificity of 0. So the most valued attribute among the models will be a balanced trade-off between sensitivity, and specificity.

The ARI ranges from -1 to 1, where a score of 1 indicaties that the label distribution of two groupings are perfectly matched, a score close to 0 indicates that there is very little overlap between label distribution of the two groupings and a score close to -1 indicates that the overlap of the labels of two groupings is worse than what would be expected by two sets of random label distributions. The main strength of the ARI is also the reason why it is used by many authors to evaluate clustering models CITATION, it allows for the comparison of a set of cluster assignments where the number of clusters is greater than the number of labels.

Chapter 7

Results

7.1 Case Study: Heart Failure

7.1.1 Time-series Clustering

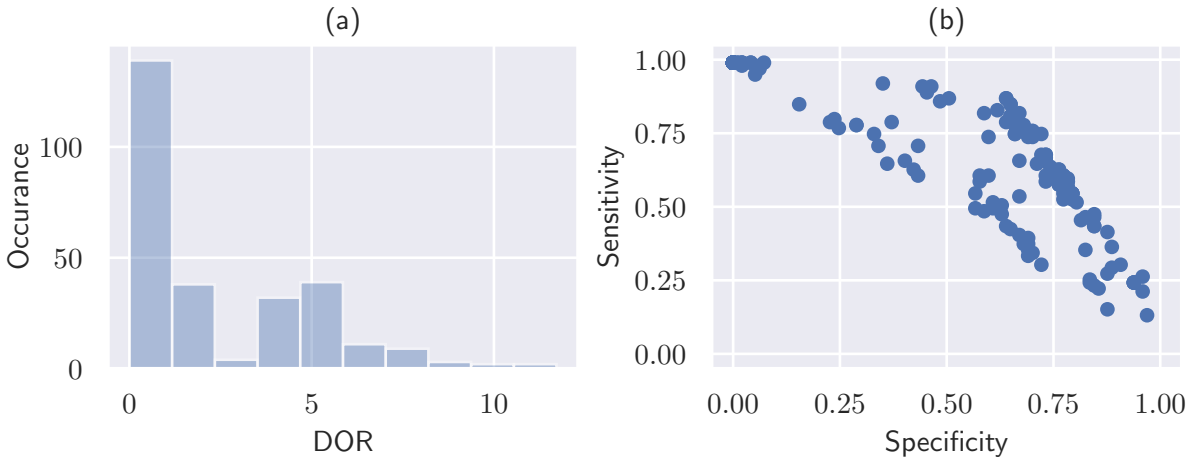


Figure 7.1: (a) Distribution plot of DOR of all TSC models evaluated at two cluster centers when applied to classify heart failure. (b) Scatter plot of the same models sensitivity, and specificity.

Figure 7.1a shows that the DOR is close to zero for many of the two-cluster-center models, However, the best performing models are able to acheive a DOR above ten, these models are listed in table 7.1. From the scatterplot in figure 7.1b one can see that the distribution of sensitivity, and specificity are quite widespread. Sensitivity and specificity scores range from 0 to 1. Common to the top 18 models in terms of DOR is that they all use data from a single view, and 2CH is the only view that is represented among the five models with highest DOR. What else is worth noting is that almost all the models using normalization or z-normalization as preprocessing score below the models that use scaling, or no preprocessing at all. These observations can be confirmed from the table10.1 in the appendix. From table 7.1 one can see that the two best-performing models in terms of DOR received the exact same score in all metrics. *gls/2CH/regular/centroid/2*, and *gls/2CH/scaled/centroid/2* differ only in the way of preprocessing, the former does not preprocess the curves before clustering, and the latter uses scaling. However, for these two cases preprocessing did not matter as they have the exact same cluster assignments as well.

Dataset-model	Accuracy	Sensitivity	Specificity	DOR
GLS/2CH/regular/centroid/2	0.76	0.87	0.64	11.72
GLS/2CH/scaled/centroid/2	0.76	0.87	0.64	11.72
GLS/2CH/regular/average/2	0.75	0.85	0.65	10.38
GLS/2CH/scaled/average/2	0.75	0.85	0.65	10.38
GLS-rls/2CH/scaled/ward/2	0.74	0.82	0.67	9.14

Table 7.1: The accuracy, DOR, sensitivity and specicity scores of the five best performing two-cluster-centerTSC models in terms of DOR, at detecting heart failure. The **Dataset-model** column indicates *Dataset used/ View used/ Type of preprocessing used/Linkage criteria of model/Number of cluster centers.*

Dataset-model	ARI
GLS/2CH/regular/centroid/2	0.25
GLS/2CH/scaled/centroid/2	0.25
GLS/2CH/scaled/centroid/3	0.24
GLS/2CH/regular/centroid/3	0.24
GLS/2CH/scaled/average/2	0.24

Table 7.2: The five highest ARI scores attained when applyingTSC for detecting heart failure. The **Dataset-model** column indicates *Dataset used/ View used/Linkage criteria of model/Number of cluster centers.*

The majority of ARI scores are close to zero, but 17 models evaluated at different numbers of cluster centers are able to acheive an ARI score above 0.20. As with DOR, the general trends for models with a high ARI score is that they use data from a single view, use scaling or no preprocessing at all. From table 7.2 one can see that the top five models only use the GLS curve from the 2CH view. In addition, one can also see that the two models with the highest ARI (0.25) are the clustering models evaluated at two cluster centers that perform best in terms of DOR as well. This means that there most likely are no models evaluated at a number of cluster centers higher than two that will perform better than *gls/2CH/regular/centroid/2*, or *gls/2CH/scaled/centroid/2*. Figure 7.2 shows the 2CH GLS curves of five random cluster members from the *gls/2CH/regular/centroid/2* model. Although one caANNot make any conclusive statements about what the general similarities between cluster members are, from the plots in figure 7.2 it seems like the curves of cluster 2 are smooth, while the curves of cluster 1 are more irregular in shape, which makes sense as this clustering algorithm uses a shape-based distance measure. Since *gls/2CH/regular/centroid/2* is one of two models to acheive the highest DOR (11.72), accuracy (0.76), and ARI (0.25) it is chosen as the best of theTSC models at identifying heart failure among patients. *gls/2CH/regular/centroid/2* is chosen over *gls/2CH/scaled/centroid/2* because it does not require preprocessing.

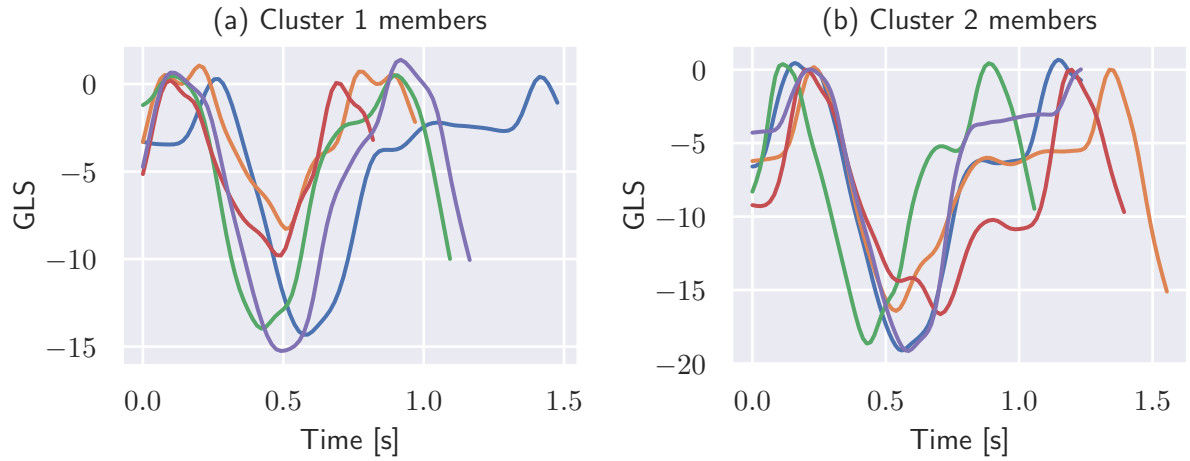


Figure 7.2: Here the curves of five random cluster members assigned by the *gls/2CH/regular/centroid/2* model. Each plot depicts the 2CH GLS curves for five random cluster members from the *gls/2CH/regular/centroid/2* model. (a) and (b) contain members from cluster 1 and 2 respectively. Only five curves are included to avoid making the plot to chaotic.

7.1.2 Peak-value Clustering

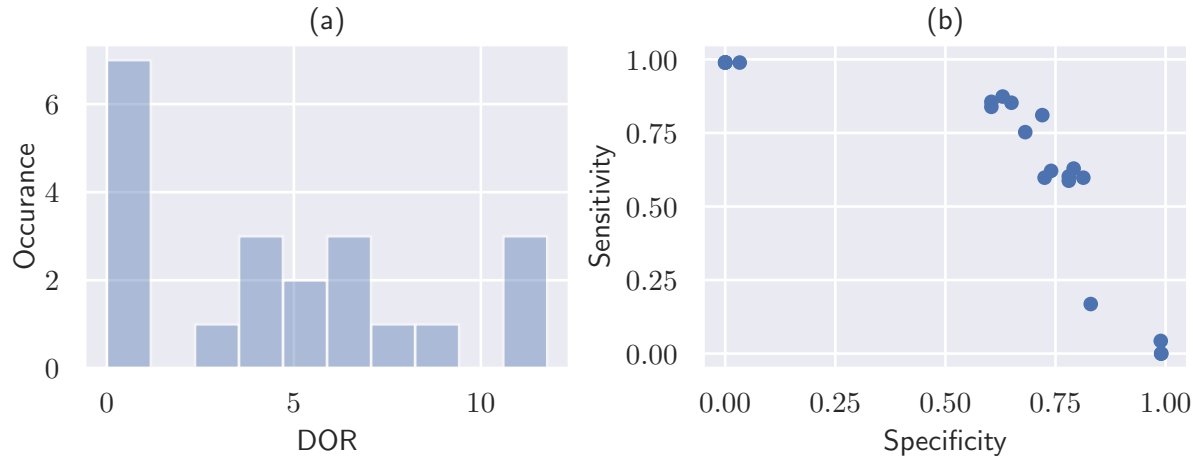


Figure 7.3: (a) Distribution plot of DOR of all PVC models evaluated at two cluster centers when applied to classify heart failure. (b) Scatter plot of the same models sensitivity, and specificity.

From figure 7.3a one can see that the majority of DOR scores are centered around zero, but there is a substantial number of models that acheive a DOR score above 10. The scatterplot in figure 7.3b shows that there is also a great spread in sensitivity, and specificity. A few models are spread along the edges of the plot acheiving a sensitivity or specificity score close to zero, but there are also models that aceive sensitivity and specificity scores above 0.7. Common to the highest performing PVC models is that they all use the dataset that is a combination of peak systolic GLS values and EF values. This can be confirmed from the complete table of results in the appendix 10.4. From table 7.3 one can see that *gls-EF/ward/2* is the PVC model that acheives the highest DOR of 11.59 when applied to classify heart failure. The *gls-EF/complete/2* model acheives the second highest DOR of 10.85, but its' specificity is nine points higher than

gls-EF/ward/2, while its sensitivity is only six points lower, and it also has the highest accuracy of all the PVC models applied to identify heart failure.

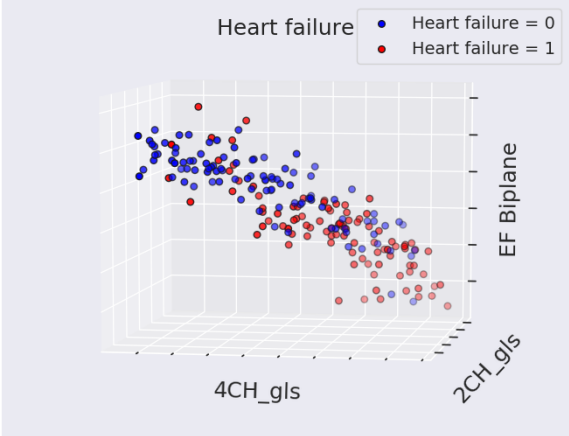
Dataset-model	Accuracy	Sensitivity	Specificity	DOR
gls-EF/ward/2	0.75	0.87	0.63	11.59
gls-EF/complete/2	0.76	0.81	0.72	10.85
gls-EF/average/2	0.75	0.85	0.65	10.58
rls-EF/complete/2	0.73	0.86	0.60	8.89
gls-rls-EF/ward/2	0.72	0.84	0.60	7.80

Table 7.3: The accuracy, DOR, sensitivity and specicity scores of the five best performing two-cluster-center PVC models in terms of DOR, at detecting heart failure. The **Dataset-model** column indicates *Dataset used/Linkage criteria of model/Number of cluster centers*.

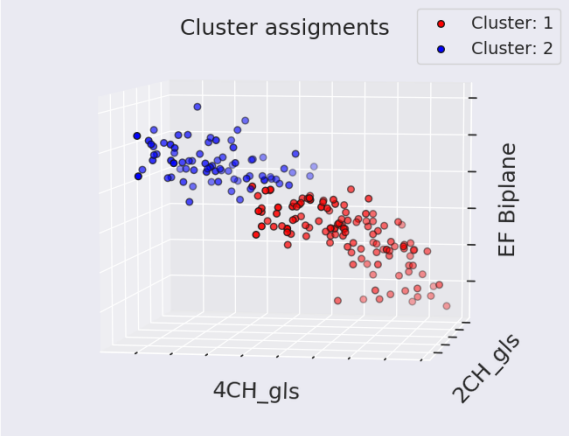
Dataset-model	ARI
gls-EF/complete/2	0.27
gls-EF/ward/2	0.24
gls-EF/average/2	0.24
rls-EF/complete/2	0.21
gls-EF/complete/3	0.21

Table 7.4: The five highest ARI scores attained when applying PVC for detecting heart failure. The **Dataset-model** column indicates *Dataset used/Linkage criteria of model/Number of cluster centers*.

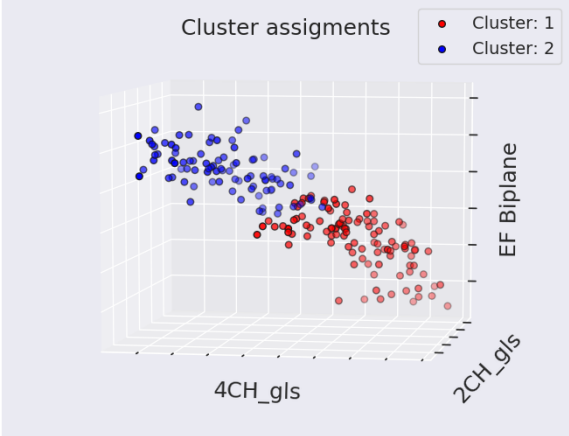
Many of the ARI of PVC models for classifying heart failure are close to zero, but substantially more of the models score above zero in ARI As with DOR, the models that acheive the highest ARI scores use datasets that are combinations of strain curves and EF values. Table 7.4 shows that the three highest ARIs are attained by the same three models that acheived the highest DORs. This means that there are most likely no models evaluated at a higher number of cluster centers that will outperform *ward/2*, or *complete/2* at classifying heart failure. However, *complete/2* acheives the highest ARI, although it only acheives the second highest DOR. *complete/2* is chosen as the best performing PVC model when classifying heart failure, since it has the highest accuracy (76%), highest ARI (0.27), and second highest DOR (10.85). In figure 7.4 scatterplots patients are plotted with the dimensions: 4-chamber peak systolic GLS, 2-chamber peak systolic GLS and EF. The colors of the points correspond to wheather the patient has heart failure or not, and which cluster the points belong to. The plots are actually a lower dimensional projection of the GLS-EF peak-value dataset. This particular projection was chosen as it was found to be the projection where heart failure patients were as separable as possible. From plots 7.4b-d one can see that the clusters are fairly separable, heart failure on the other hand is not as easy to separate in these dimensions as can be seen in plot 7.4d. *Ward/2* and *complete/2* can in some sense be considered as binary classifiers where values under a certain threshold are categorized as heart failure. The *ward/2* model has the highest threshold for what is considered heart failure, and *complete/2* has the lowest, which explains their difference in sensitivity and specificity score. Since model *complete/2* acheives the highest accuracy (0.76), highest ARI (0.27) and second highest DOR (10.85) it is chosen as the best PVC model to identify heart failure among patients.



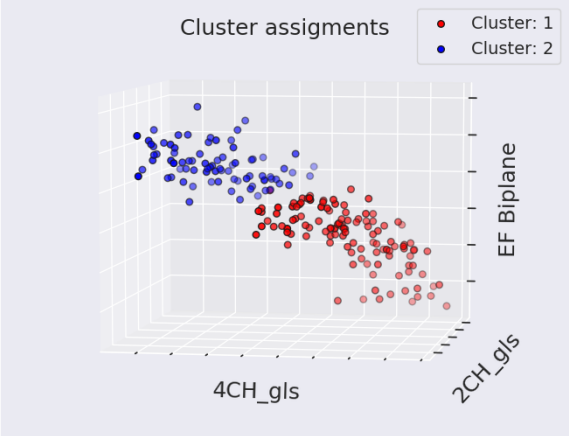
(a) Heart failure.



(b) *Ward/2* cluster assignments.



(c) *Complete/2* cluster assignments.



(d) *Average/2* cluster assignments.

Figure 7.4: Scatterplot of peak GLS values in each view. Colors in the of the different dots are given by heart failure diagnosis, and cluster assignments of ward/2, complete/2 and average/2 models. Numbers are not included on the axes because the point of the figure is to illustrate the separability of clusters, and heart failure.

Dataset-Model	Accuracy	Sensitivity	Specificity	DOR
gls/4CH/upsampled	0.54	0.46	0.61	1.36
rls/APLAX/regular	0.53	0.48	0.58	1.30
rls/4CH/regular	0.52	0.36	0.68	1.20
gls/APLAX/downsampled	0.52	0.63	0.40	1.15
gls/2CH/downsampled	0.51	0.61	0.40	1.03

Table 7.5: The accuracy, DOR, sensitivity and specicity scores of the five best performing variations of the ANN in terms of DOR, at detecting heart failure. The **Dataset-Model** column indicates *Dataset used/ View used/ Whether curve has been upsampled, downsampled or is regular.*

7.1.3 Deep Neural Network

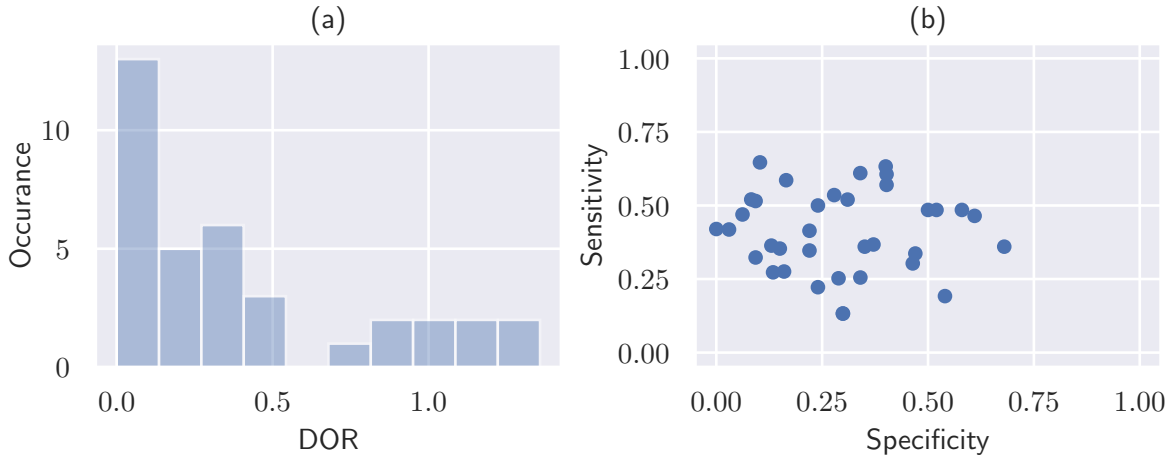


Figure 7.5: (a) Distribution plot of DOR of all ANN models evaluated at two cluster centers when trained to predict heart failure. (b) Scatter plot of the same models sensitivity, and specificity.

From the distribution plot in figure 7.5a one can see that the most frequent DOR by ANN models when training them to predict heart failure is zero . The highest DOR of 1.36 is attained by using only the GLS curve from the 4CH view as input, as can be seen from table 7.12. In the scatterplot in figure 7.5b one can see that sensitivity scores vary between 0.15 and 0.65, and the specificity scores vary between 0 and 0.68. The majority of the ANN variations acheive a sensitivity, specificity and accuracy below 0.50. The accuracy of the model variations are also fairly low, 0.54 being the highest accuracy acheived. Since the heart failure dataset is fairly evenly distribution (recall figure 5.7) an accuracy of 0.54 is not much better than what could be acheived by randomly guessing the label. The 11 highest DORs attained by ANN models trained to classify heart failure are acheived using only curves from single views as input, and only GLS, or RLS curves. *Gls/4CH/upsampled* will be considered the best model variation of the ANNs at predicting heart failure since it acheives the highest accuracy and DOR .

Dataset-Model	Accuracy	Sensitivity	Specificity	DOR
gls-EF/Gaussian-Process	0.75	0.78	0.73	9.40
rls-EF/MLP	0.75	0.76	0.74	9.37
rls-EF/Linear-SVM	0.75	0.75	0.74	8.86
gls-EF/Ada-Boost	0.75	0.77	0.73	8.85
gls-EF/Naive-Bayes	0.75	0.76	0.74	8.79

Table 7.6: The accuracy, DOR, sensitivity and specicity scores of the five best performing PVSC in terms of DOR, at detecting heart failure. The **Dataset-Model** column indicates *Dataset used/The specific ML model used.*

7.1.4 Peak-value Supervised Classifiers

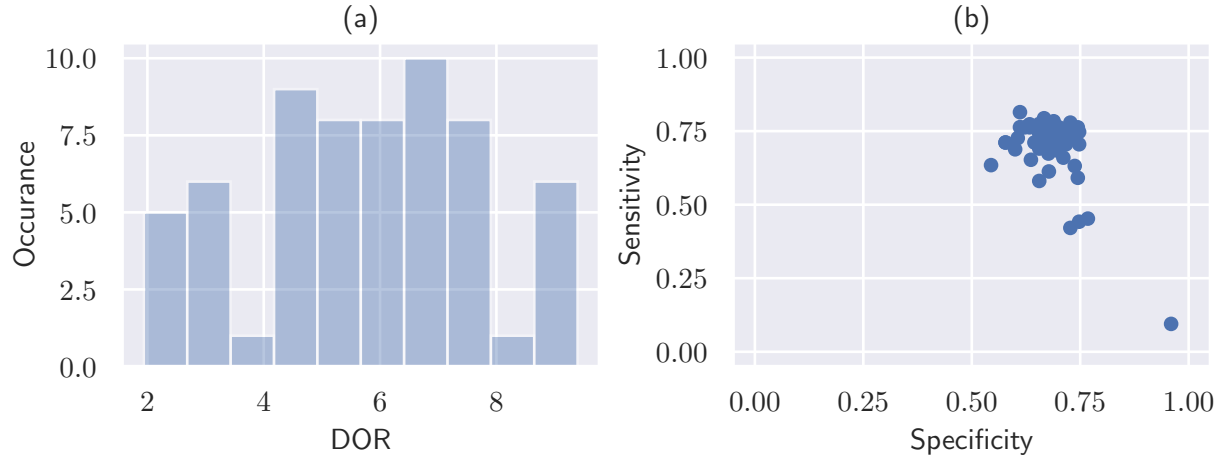


Figure 7.6: (a) Distribution plot of DOR of all PVSC models evaluated at two cluster centers when trained to predict heart failure. (b) Scatter plot of the same models sensitivity, and specificity.

From the distribution plot depicted in figure 7.6a one can see that the PVSC models overall acheive relatively high DORs, with a range of approximately two to nine. The scatterplot in figure 7.6b shows that the models are quite concentrated in terms of sensitivity and specificity scores. The majority of the models achieve sensitivity, and specificity scores in the ranges 0.6 to 0.75, with some outliers acheiving specificity below 0.5 and sensitivity above 0.75. What is even more concentrated are the accuracy scores of the models. As can be seen in table 7.6, the accuracy of top five PVSC models are all 0.75 As with PVC all the best performing PVSC models use a combination of EF and peak systolic strain values, and no specific ML model seems to outperform the others on all the datasets in term of DOR. The table also shows that the highest DOR of 9.4 is acheived by model *gls-EF/Gaissian-Process*. Although the DOR, sensitivity and specificity scores are very similar for the five best performing models *gls-EF/Gaussian-Process* is chosen as the PVSC model that performs best at predicting heart failure as it acheives the highest DOR.

Dataset-Model	Accuracy	Sensitivity	Specificity	DOR
TSC-gls/2CH/regular/centroid/2	0.76	0.87	0.64	11.72
PVC-gls-EF/complete/2	0.76	0.81	0.72	10.85
ANN-gls/4CH/upsampled	0.54	0.46	0.61	1.36
PVSC-gls-EF/Gaussian-Process	0.75	0.78	0.73	9.40
Dataset-Model	TP	TN	FP	FN
TSC-gls/2CH/regular/centroid/2	86	62	35	13
PVC-gls-EF/complete/2	77	72	28	18
ANN-gls/4CH/upsampled	46	61	39	53
PVSC-gls-EF/Gaussian-Process	74	72	27	21

Table 7.7: A table comparing the best contenders within each model group for predicting heart failure among patients. The top table comprare the models by their accuracy, sensitivity, specificity and DOR, and the bottom table shows the number of TPs, TNs, FPs and FNs that the different models attain.

7.1.5 Comparisons

With exeption of the ANN, the models performance of the different models are very close in terms of DOR and accuracy. From table 7.7 one can see that theTSC model *gls/2CH/regular/centroid/2* achieves the highest sensitivity of all the models applied to predict heart failure, but it achieves the second lowest specificity of the four model groups. This can be confirmed by the fact that it attains 86 TPs, and 35 FPs. The PVSC model *gls-EF/Gaussian-Process* attains the most balanced score in terms of sensitivity and specificity, and the highest specificity score of all the model groups. However, the PVC model *gls-EF/complete/2* attains a higher accuracy, sensitivity and DOR than the PVSC model. One can also see that the PVC model attains more TP, the same number of TN, fewer FP and fewer FN than the PVSC model. It should also be noted that the PVC model and the PVSC model are using the same dataset which is a combination of peak systolic GLS values, and EF. To conclude this particular case study, the PVC model is picked as the best model at predicting heart failure among patients as it achieves the highest accuracy of the model groups, highest number of TN, and one of the most balanced combinations of sensitivity, and specificity.

Dataset-model	Accuracy	Sensitivity	Specificity	DOR
gls/2CH/regular/centroid/2	0.74	0.71	0.93	33.47
gls/2CH/scaled/centroid/2	0.74	0.71	0.93	33.47
gls/2CH/scaled/average/2	0.73	0.69	0.93	30.71
gls/2CH/regular/average/2	0.73	0.69	0.93	30.71
gls/2CH/scaled/ward/2	0.71	0.67	0.93	27.49

Table 7.8: The accuracy, DOR, sensitivity and specicity scores of the five best performing two-cluster-center TSC models in terms of DOR, at detecting patient diagnoses. The **Dataset-model** column indicates *Dataset used/ View used/ Type of preprocessing used/ Linkage criteria of model/ Number of cluster centers.*

7.2 Case Study: Patient Diagnosis

7.2.1 Time-series Clustering

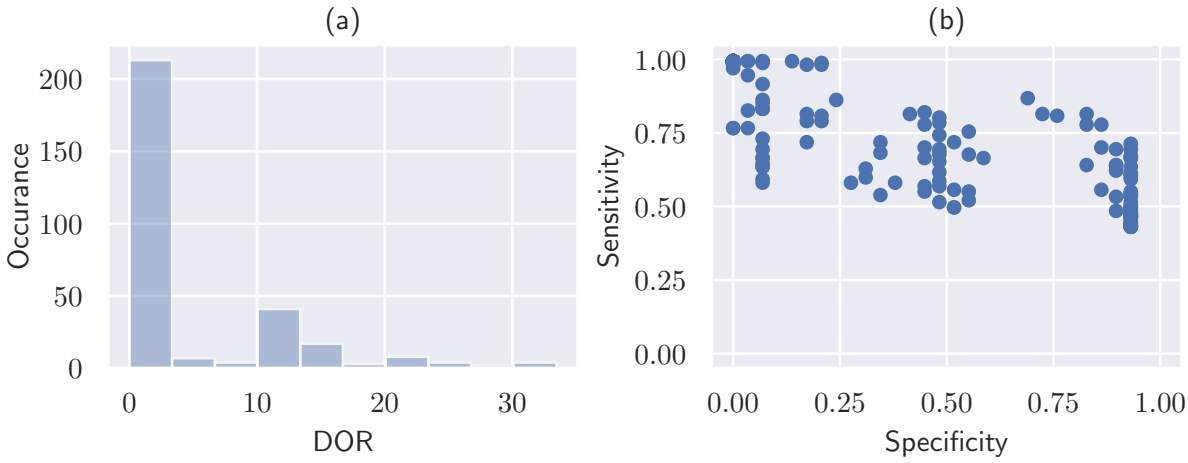


Figure 7.7: (a) Distribution plot of DOR of all TSC models evaluated at two cluster centers when applied to classify patient diagnosis. (b) Scatter plot of the same models sensitivity, and specificity.

From the distribution plot in figure 7.7a one can see that the majority of DORs are close to zero, but there are some models that acheive a DOR above 30. In the scatter plot in figure 7.7b one can see that the specificity of the models and range from 0.5 to 1, and the sensitivity scores range from 0 to 0.93. As with heart failure, the TSC models that perform best in terms of DOR use data from a single view. The 2CH view, and GLS curves are the only view and curve that are used among the models that achieve the five highest DORs. From the table of all the model results in the appendix 10.2 one can see that the highest performing model in terms of DOR to use a dataset other than GLS curves alone is *gls-rls/2CH/scaled/ward/2* and it achieves a DOR of 26.76. One can also note that the highest performing model in terms of DOR that uses a view other than only 2CH is *rls/all-views/normalized/weighted/2* which achieves a DOR of 25.56. The TSC models that achieve the highest DOR scores all use no preprocessing, or scaling. From table 7.8 one can see that the TSC models that acheive the highest DOR scores are *gls/2CH/regular/centroid/2*, and *gls/2CH/scaled/centroid/2* which are the same two models that achieve the highest DORs in the heart failure case study.

Dataset-model	ARI
gls-rls/4CH/regular/complete/2	0.36
gls/all-views/regular/weighted/2	0.34
gls/all-views/scaled/weighted/4	0.33
gls/all-views/scaled/weighted/3	0.33
gls/APLAX/regular/single/10	0.32

Table 7.9: The five highest ARI scores attained when applying TSC for detecting patient diagnoses. The **Dataset-model** column indicates *Dataset used/ View used/ Linkage criteria of model/ Number of cluster centers.*

The majority of the ARI scorer for all the TSC models evaluated at two to nine cluster centers are centered around zero. As with the TSC models attaining the highest DORs the models using no preprocessing or scaling acheive the highest ARI indices when used to identify patient diagnoses. In addition, the GLS curves are also most often part of the dataset for the TSC models receiving the highest ARI when used to identify patient diagnoses. From table 7.9 one can see that the TSC models receiving the five highest ARI scores, are not among the TSC models that receive the highest DOR scores. The TSC model *gls-rls/4CH/regular/complete/2* attains the highest ARI score when applied to identify patient diagnoses, and achieves an accuracy of 0.84, a sensitivity of 0.87 a specificity of 0.69 and a DOR 14.65. The TSC model *gls/all-views/regular/weighted/2* achieves the second highest ARI when applied to identify patient diagnoses, and achieves an accuracy of 0.82, a sensitivity of 0.81 a specificity of 0.83 and a DOR 21.06. What should also be noted is that the TSC models achieving the two highest ARIs when applied to identify patient diagnoses are models evaluated at two cluster centers, which means that none of the TSC models evaluated at cluster centers between three and nine can perform better than the ones evaluated at two cluster centers. It may seem strange that the ordered lists of DORs, and ARIs are so different. The reason for this is not because DOR inherently values sensitivity higher than specificity, but stems from how the DOR is defined. Recall that $\text{DOR} = (\text{TP} \times \text{TN})/(\text{FP} \times \text{FN})$, since the patient diagnoses dataset is skewed in favour of positives TP has the potential of being as high as 170 while TN can be as high as 30. Therefore the DOR will be higher for models with a high sensitivity than for models with an equally high sensitivity. In figure 7.8 curves of five random cluster members assigned by the *gls/all-views/regular/weighted/2* model are plotted. As with the observations made with regard to figure 7.2 it is not possible to make any conclusive statements as to what the similarities are based on such a small sample size. However, based on the small sample size in 7.8 it seems as though the curves in cluster 2 (column (b)) are smoother in shape, than the curves in cluster 1 (column (a)). The TSC model that is chosen as the best model for identifying patient diagnoses is *gls/all-views/regular/weighted/2*, because it achieves the second highest ARI, and because it's sensitivity and specificity are more balanced than the model attaining the highest ARI and the models that achieve higher DORs.

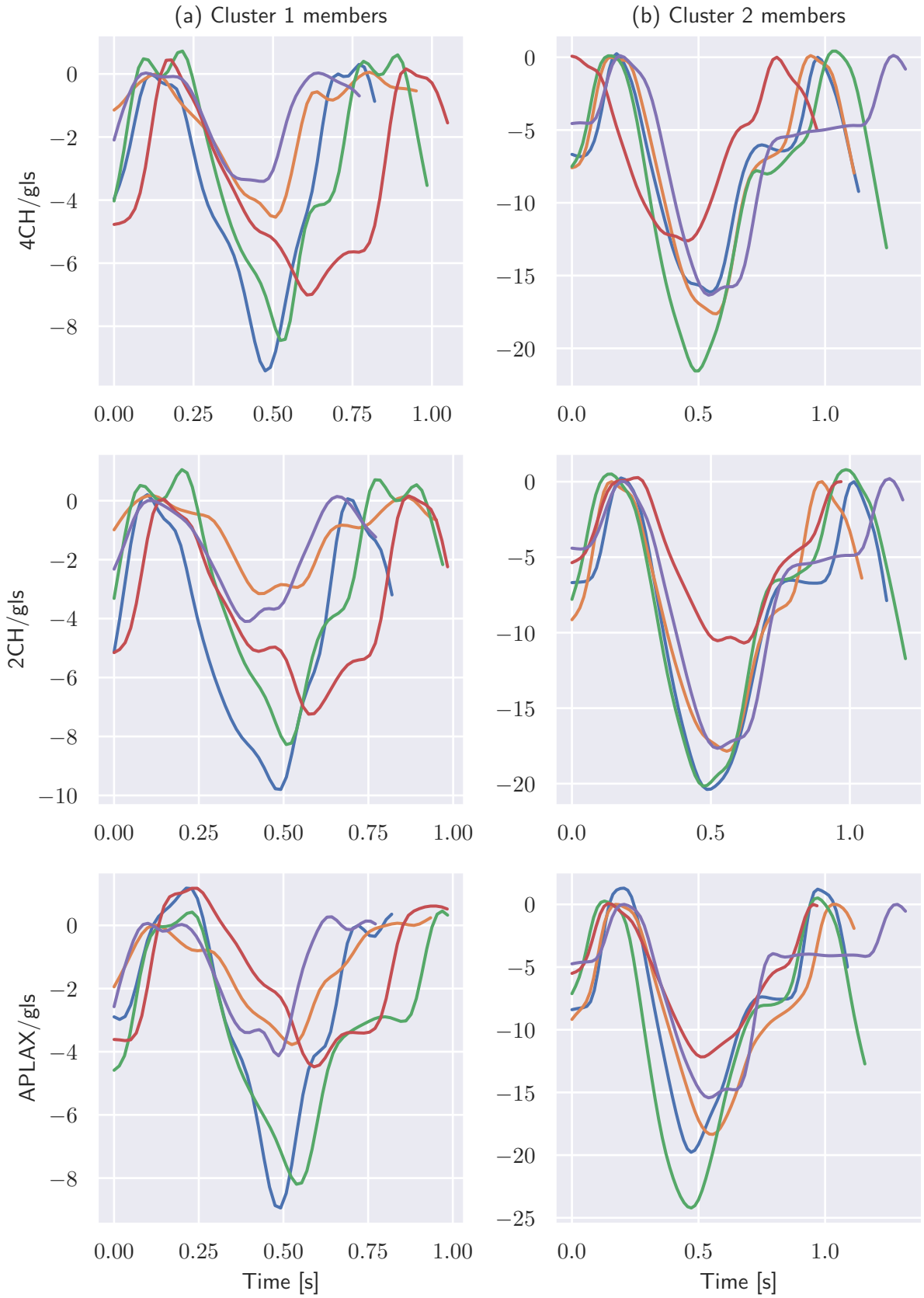


Figure 7.8: Here the curves of five random cluster members assigned by the *gls/all-views/regular/weighted/2* model are plotted. Each row represents one of the seven possible strain curves in the 4CH view. Coloumn (a) and (b) represent cluster 1 and 2 respectively. To make it easier to visually separate the curves, only five random members from cluster 1 and 2 are included in the figure.

Dataset-model	Accuracy	Sensitivity	Specificity	DOR
gls-EF/ward/2	0.76	0.72	0.94	39.33
rls-EF/complete/2	0.77	0.74	0.93	37.61
gls-rls-EF/ward/2	0.76	0.72	0.93	35.16
gls-EF/average/2	0.74	0.70	0.94	34.90
gls-EF/complete/2	0.68	0.63	0.94	25.75

Table 7.10: The accuracy, DOR, sensitivity and specicity scores of the five best performing two-cluster-center PVC models in terms of DOR, at detecting patient diagnoses. The **Dataset-model** column indicates *Dataset used/Linkage criteria of model/Number of cluster centers.*

7.2.2 Peak-value Clustering

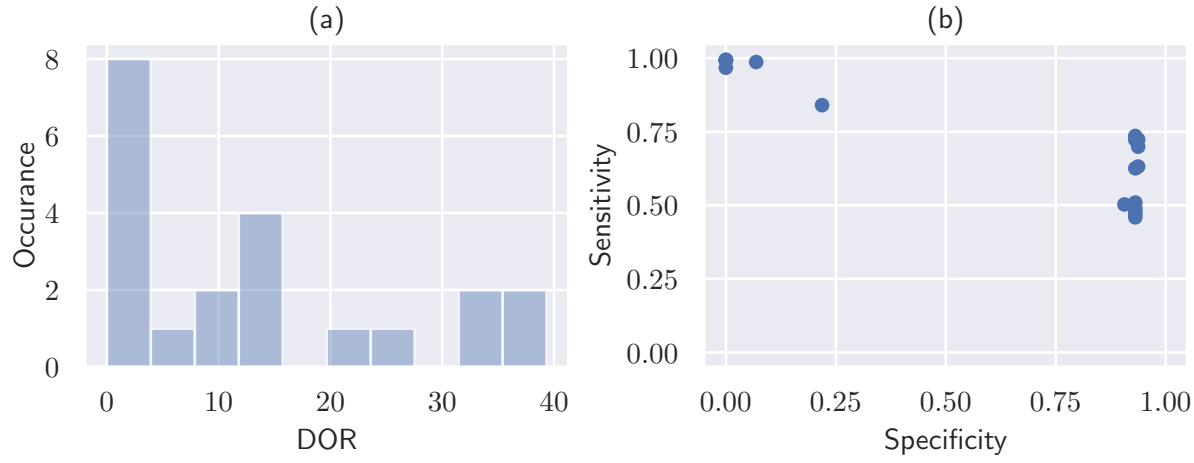


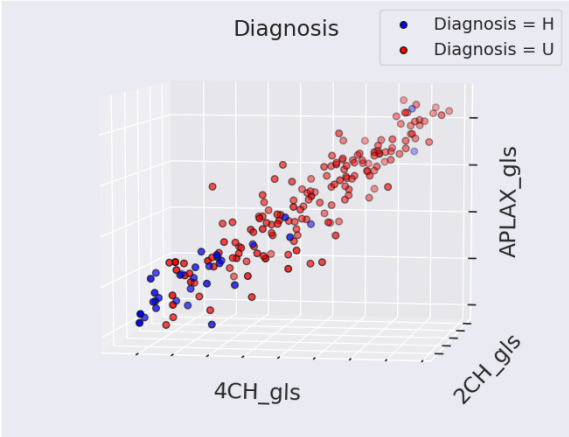
Figure 7.9: (a) Distribution plot of DOR of all PVC models evaluated at two cluster centers when applied to classify patient diagnosis. (b) Scatter plot of the same models sensitivity, and specificity.

From the distribution plot in figure 7.9a one can see that the majority of the PVC models get DORs close to zero, but there are a few models that attain DORs above 30, and close to 40. From the scatter plot in 7.9b one can see that almost all the sensitivity scores are above 0.5, while the specificity scores are concentrated in the areas 0 to 0.25 and 0.95. As with the heart failure case study the PVC models that perform high in terms of DOR use a dataset that is a combination of peak systolic strain values and EF. From table 7.10 one can see that *gls-EF/ward/2* and *rls-EF/complete/2* are the two top performers in terms of DOR. *gls-EF/ward/2* achieves a slightly higher specificity score, where as *rls-EF/complete/2* attains a slightly higher specificity score.
The majority of the ARI scores of PVC models applied to identify patient diagnoses are centered around zero, but as one can see from table 7.11 there are a few models that acieve an ARI above 0.2 close to 0.3. For a change, the PVC models that perform best in terms of ARI, are neither models evaluated at two cluster centers, or models that are applied on a combination of peak systolic strain values and EF. In contrast to the heart failure case study, the PVC models that achieve the highest ARIs, when applied to identify patient diagnoses, are not the same models that achieve the highest DORs. The two PVC models that achieve the highest ARIs are the *gls/average* model evaluated at 6 and 7 cluster centers respectively. To get a better idea of why

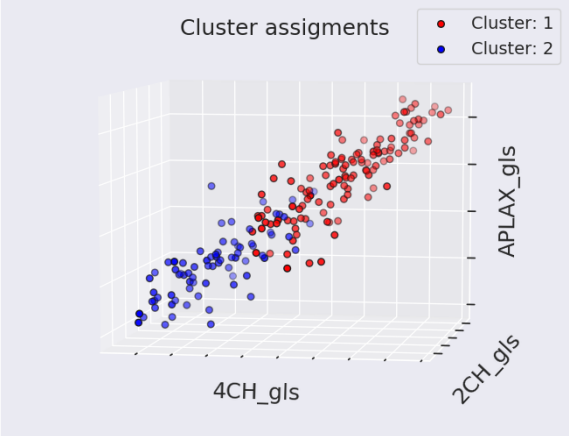
Dataset-model	ARI
gls/average/6	0.29
gls/average/7	0.29
gls-rls/complete/3	0.28
rls-EF/complete/2	0.26
gls-EF/ward/2	0.25

Table 7.11: The five highest ARI scores attained when applying PVC for detecting patient diagnoses. The **Dataset-model** column indicates *Dataset used/Linkage criteria of model/Number of cluster centers.*

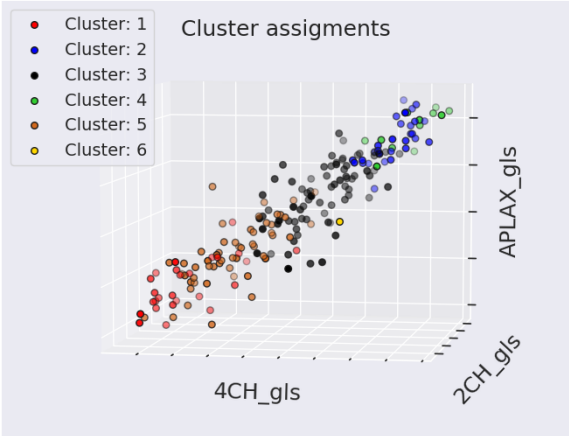
gls/average/6 and *gls/average/7* attain the ARIs they do, scatter plots of these two models, and *gls-EF/ward/2* have been given in figure 7.4. A scatter plot of the target variable patient diagnosis is also given for comparison. The dimensions used are peak systolic GLS in all three views as these are the dimensions that are common to all three models. From the scatter plot in plot 7.10a one can see that the healthy patients are in the minority, and are concentrated in the corner with low peak systolic GLS values in the 4CH, 2CH and APLAX views. There are also some healthy patients with low-medium peak systolic GLS values, and very few healthy patients with high peak systolic GLS values. From plot 7.10b one can see that *gls-EF/ward/2* is able to isolate the concentration of healthy patients with low peak systolic GLS, but at the cost of many FNs. * In plot 7.10c and 7.10d one can see that cluster 1 of model *gls/average/6*, and cluster 2 of model *gls/average/7* capture the healthy patients with low peak systolic GLS, but are unable of capturing the healthy patients with medium to high values. If one combines clusters 1 and 5 of *gls/average/6*, and lets them represent healthy patients, and let the remaining clusters represent unhealthy patients the model attains an accuracy of 0.74, a sensitivity of 0.70, a specificity of 0.94 and a DOR of 34.90. If one combines clusters 2 and 5 of *gls/average/7*, and lets them represent healthy patients, and let the remaining clusters represent unhealthy patients this model attains an accuracy of 0.74, a sensitivity of 0.70, a specificity of 0.94 and a DOR of 35.94. While the performance of the revised *gls/average/6* and *gls/average/6* models are good, they are still not as good as the performance of the top three performers in terms of DOR, which attain higher accuracy, sensitivity and DORs. Therefore, *rls-EF/complete/2* is chosen as the best of the PVC models at identifying patient diagnosis, as it achieves the second highest DOR, and a more balanced sensitivity/specificity than *gls-EF/ward/2* that attains the highest DOR score.



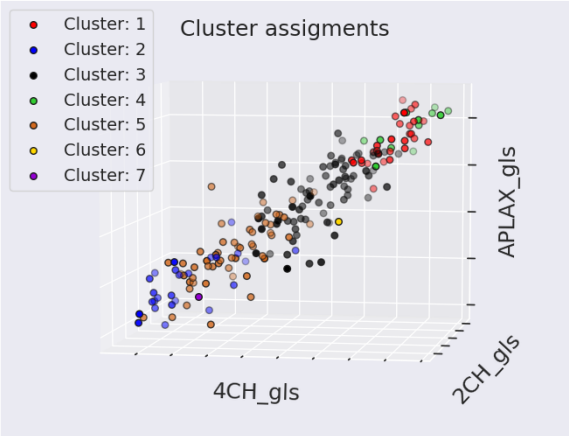
(a) Patient Diagnosis. **H** stands for **Healthy**, and **U** stands for **Unhealthy**



(b) *GLS-EF Ward/2* cluster assignments.



(c) *GLS Average/6* cluster assignments.



(d) *GLS Average/7* cluster assignments.

Figure 7.10: Scatterplot of peak GLS values in each view. Colors in the of the different dots are given by heart failure diagnosis, and cluster assignments of *gls-EF/ward/2*, *average/6* and *average/7* models. Numbers are not included on the axes because the point of the figure is to illustrate the separability of clusters, and patient diagnosis.

Dataset-Model	Accuracy	Sensitivity	Specificity	DOR
all-strain/4CH/upsampled	0.83	0.99	0.00	0.00
all-strain/2CH/regular	0.85	1.00	0.00	NaN
gls/2CH/regular	0.85	1.00	0.00	NaN
rls/2CH/regular	0.85	1.00	0.00	NaN
all-strain/2CH/downsampled	0.85	1.00	0.00	NaN

Table 7.12: The accuracy, DOR, sensitivity and specicity scores of the five best performing variations of the ANN in terms of DOR, when trained to predict patient diagnoses. The **Dataset-Model** column indicates *Dataset used/ View used/ Whether curve has been upsampled, downsampled or is regular.*

7.2.3 Deep Neural Network

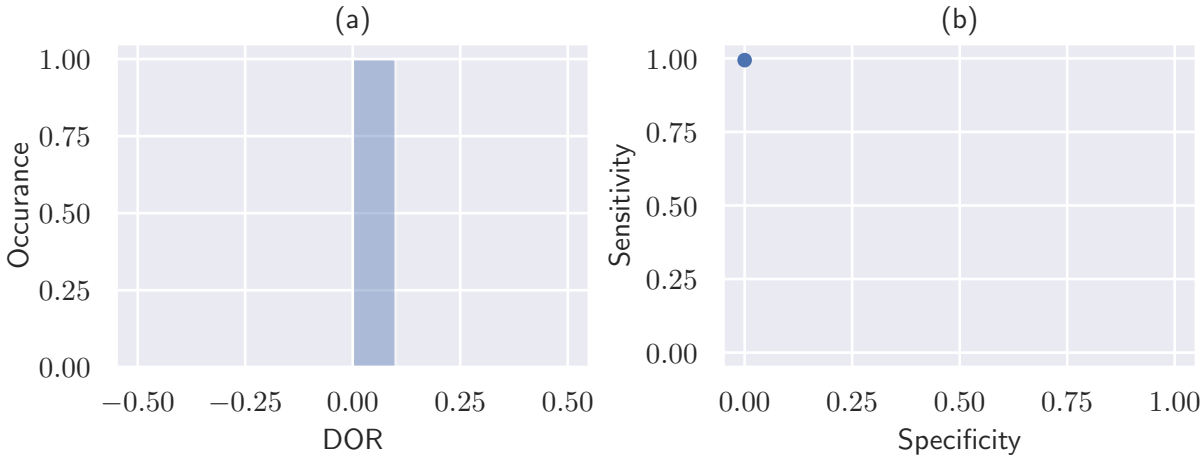


Figure 7.11: (a) Distribution plot of DOR of all ANN models when trained to classify patient diagnosis. (b) Scatter plot of the same models sensitivity, and specificity.

From the distribution plot in figure 7.11 one can see that the collective performance of the different variations of the ANN trained to predict patient diagnosis is terrible. The DOR of all the models are either zero because the number of TNs attained are zero, or not defined because the number of FNs are zero. The sensitivities are all 1, or close to 1, and the specificities are all 0. It is evident that the ANNs are not able to generalize the traits of the healthy patients from such a small dataset. The ANN models are will therefore not be discussed further with relation to prediction of patient diagnosis, and are not included in the comparison of the four model groups.

Dataset-Model	Accuracy	Sensitivity	Specificity	DOR
gls-rls-EF/Ada-Boost	0.95	0.97	0.79	138.42
gls-rls/KNN	0.93	0.95	0.82	84.53
rls-EF/Extra-Trees	0.93	0.96	0.75	76.50
gls-rls-EF/Extra-Trees	0.93	0.97	0.71	75.00
gls-rls/Extra-Trees	0.93	0.97	0.71	75.00

Table 7.13: The accuracy, DOR, sensitivity and specicity scores of the five best performing PVSC models in terms of DOR, when trained to predict patient diagnosis. The **Dataset-Model** column indicates *Dataset used/Specific machine learning model used.*

7.2.4 Peak-value Classifiers

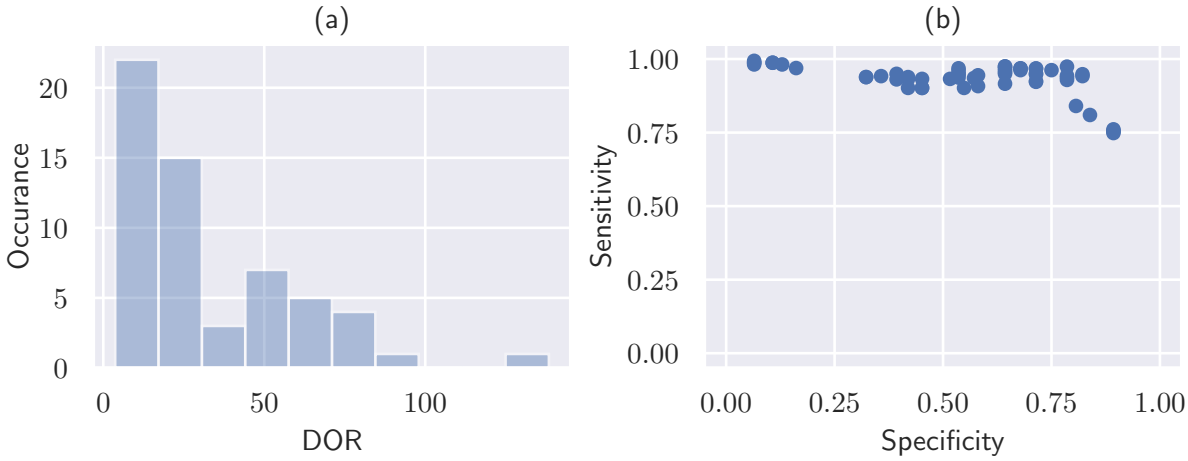


Figure 7.12: (a) Distribution plot of DOR of all PVSC models when trained to classify patient diagnosis. (b) Scatter plot of the same models sensitivity, and specificity.

From the distribution plot in figure 7.12 it might seem like the majority of the DOR scores are close to zero, but in that is due to the shear spread of DOR scores so it should be said explicitly that the lowest DOR score of a PVSC model is 3.68 and is attained by the *gls/Gaussian-Process* model. The spread of DOR is so great that some models attain a DOR close to 100, and one model attains a DOR close to 150. From the scatter plot in figure 7.12 one can see that the sensitivity ranges from 0.75 to 1, and the specificity ranges from close to zero to approximately 0.95. Among the top five PVSC models in terms of DOR are many different combinations of models, and datasets. Three of the five highest DOR scores are attained by Extra-Trees models, and the top two scores are attained by KNN and Ada Boost classifiers. *gls-rls-EF/Ada-Boost* and *gls-rls/KNN* are the two top PVSC performers with regard to DOR. *gls-rls-EF/Ada-Boost* achieves the highest sensitivity of the two by two points, and *gls-rls/KNN* achieves the highest specificity of the two by three points. Since sensitivity and specificity is weighted equally in this study *gls-rls/KNN* is chosen as the best of the PVSC models trained to identify patient diagnoses.

Dataset-Model	Accuracy	Sensitivity	Specificity	DOR
TSC-gls/all-views/regular/weighted/2	0.82	0.81	0.83	21.06
PVC-rls-EF/complete/2	0.77	0.74	0.93	37.61
PVSC-gls-rls/KNN	0.93	0.95	0.82	84.53
Dataset-Model	TP	TN	FP	FN
TSC-gls/all-views/regular/weighted/2	136	24	5	31
PVC-rls-EF/complete/2	117	27	2	42
PVSC-gls-rls/KNN	147	23	5	4

Table 7.14: A table comparing the best contenders within each model group for predicting patient diagnoses. The top table comprare the models by their accuracy, sensitivity, specificity and DOR, and the bottom table shows the number of TPs, TNs, FPs and FNs that the different models attain on their respective datasets.

7.2.5 Comparisons

From the top table in 7.14 one can see that there is a significant difference in performance between the three models included for comparison. The TSC model *gls/all-views/regular/weighted/2* attains the second highest accuracy, sensitivity and specificity of the three models, but also attains the lowest DOR. The TSC model can also be said to attain the most balanced scores in terms of sensitivity and specificity. The PVC model *rls-EF/complete/2* attains the highest specificity, second highest DOR, but lowest sensitivity and accuracy of the three models. The PVSC model *gls-rls/KNN* attains the highest accuracy, sensitivity and DOR of all the models, but it also achieves the lowest specificity of all the models. However, since the PVSC model is so close to the TSC model in terms of specificity, and is so much better than the other two models in all other metrics, it is chosen as the best model of identifying patient diagnoses. This can be confirmed from the bottom table in 7.14, where one can see that the PVSC model only gets one TN less than the TSC model, but attains 11 more TP.

7.3 Case Study: Segment Indication

7.3.1 Time-series Clustering

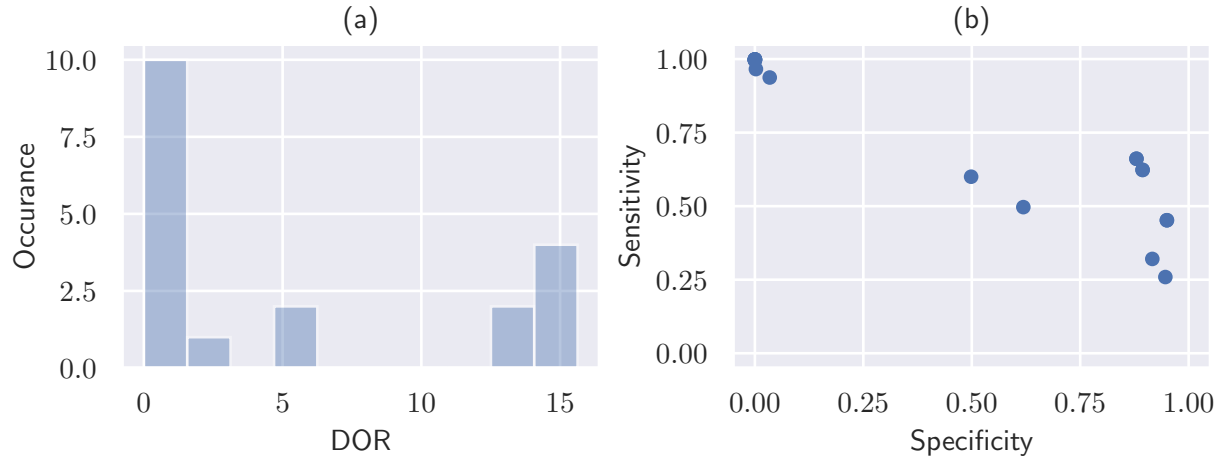


Figure 7.13: Distribution of DOR, sensitivity and specificity for the different TSC models when classifying left ventrice segment indication.

Dataset-model	Accuracy	Sensitivity	Specificity	DOR
regular/weighted/2	0.69	0.45	0.95	15.63
scaled/weighted/2	0.69	0.45	0.95	15.63
regular/ward/2	0.77	0.66	0.88	14.26
scaled/ward/2	0.77	0.66	0.88	14.26
regular/complete/2	0.75	0.62	0.89	13.92

Table 7.15: The accuracy, DOR, sensitivity and specicity scores of the five best performing two-cluster-center TSC models in terms of DOR, at detecting segment indication. The **Dataset-model** column indicates *Type of preprocessing used/Linkage criteria of model/Number of cluster centers.*

From the distribution plot in figure 7.13a one can see that the majority of the DORs are close to zero, but a few models are able to achieve DORs above 12, and some models attain a DOR close to 15 when applied to identify segment indication. From the scatter plot in figure 7.13b one can see that the sensitivity of the TSC models range from 0.25 to 1, and the specificity of the TSC models range from 0 to approximately 1. The spread in both sensitivity and specificity is quite large, and there are very few models that are able to a attain a high sensitivity while at the same time attaining a high specificity, and vice versa. Common to the high performing TSC models in terms of DOR is that they all use either no preprocessing at all, or scaling. *z-norm/complete/2* is the seventh best TSC model in terms of DOR, and attains a DOR of 5.92 when applied to identify segment indication. *norm/ward/2* is the ninth best models in terms of DOR, and attains a DOR of 1.56, when applied to identify segment indication. This can be comfirmed from table 10.3 The two TSC models attaining the highest DORs *regular/weighted/2*, and *scaled/weighted/2* differ only in type of preprocessing used. From table 7.15 and table 10.3 one can see that the two models attain the same scores in all metrics, this is because they yield the exact same cluster assignments to the individual segment strain curves. The same goes

Dataset-model	ARI
scaled/centroid/5	0.286
regular/centroid/5	0.286
regular/ward/2	0.284
scaled/ward/2	0.284
scaled/centroid/6	0.271

Table 7.16: The five highest ARI scores attained when applying TSC for detecting segmend indication. The **Dataset-model** column indicates *Type of preprocessing used/Linkage criteria of model/Number of cluster centers.*

for the next two TSC models in line *regular/ward/2 scaled/ward/2*, these two models are also the models that attain the highest accarcy of all the TSC models. Of the two TSC models *regular/weighted/2*, and *regular/ward/2* the latter is preferred for predicting segment indication because *regular/ward/2* has a more persistent performance in both sensitivity and specificity, where as *regular/weighted/2* has a high specificity, but a very low sensitivity.
The majority of the ARIs of TSC models applied to identify segment indication, but as one can see from table 7.16 some models are able to attain ARIs above 25. As with the other case studies, the TSC models that attain the highest ARIs are models that use either no preprocessing at all or scaling. Puzzlingly enough the top two TSC models for classifying segment indication in terms of ARI, are models evaluated at five cluster centers, not two. TSC models *scaled/centroid/5*, and *regular/centroid/5* differ only in type of preprocessing used, and they yield the exact same cluster assignments, and evaluations scores. The next two models in order of ARI *regular/ward/2*, and *scaled/ward/2* are familiar from the list of TSC models attaining the highest DORs when applied to identify segment indication. From table 7.16 one can also see that the difference in ARI between *regular/centroid/5*, and *regular/ward/2* is only 0.002 Since the *regular/ward/2* model will be considered the best of the TSC models at classifying segment indication. It attains the third highest ARI of all the TSC models applied to identify segment indication, and is the preferred model among the TSC models evaluated at two cluster centers.

7.3.2 Deep Neural Network

model	Accuracy	Sensitivity	Specificity	DOR
regular	0.74	0.80	0.68	8.65
downsampled	0.74	0.74	0.75	8.38
upsampled	0.65	0.55	0.73	3.36

Table 7.17: Evaluation metrics of the ANN for classifying the binary indication of individual segments in the left ventricle.

Of the three variations of the ANN model, the one that uses no resampling, and the one that downsamples all signals to the lowest sample rate achieve relatively similar DOR scores. The variation that upsamples the sample rate of all the curves to the highest sample rate performs significantly worse than the other two in terms of DOR and sensitivity. Of the three variations the model that uses downsampling is the preferred model of the three since its sensitivity and specificity are more balanced than the model that uses no resampling, and accuracy is higher than the model that uses upsampling.

7.3.3 Comparisons

From table 7.18 one can see that the performances of the ANN, and TSC models are quite close in terms of accuracy, but differ significantly in the other metrics. The TSC model *regular/ward/2* attains a higher accuracy, specificity and DOR than the ANN model *downsampled*. This can also be confirmed by the fact that the TSC model attains more TN, and fewer FP than the ANN model. The ANN model attains the highest sensitivity, which can be confirmed by the fact that it attains more TP and fewer FN than the TSC model. The ANN model is also the model that attains the most balanced scores of sensitivity and specificity. Therefore the ANN model is chosen as the best performer at predicting the segment indication.

Dataset-Model	Accuracy	Sensitivity	Specificity	DOR
TSC-regular/ward/2	0.76	0.64	0.88	13.15
ANN-downsampled	0.74	0.74	0.75	8.38
Dataset-Model	TP	TN	FP	FN
TSC-regular/ward/2	1202	1491	204	616
ANN-downsampled	1255	1390	473	440

Table 7.18: A table comparing the best contenders within each model group for predicting segment indication. The top table comprare the models by their accuracy, sensitivity, specificity and DOR, and the bottom table shows the number of TPs, TNs, FPs and FNs that the different models attain.

7.4 Chapter Summary

In the heart failure case study the PVC model was found to be the best performer, by a narrow margin. The TSC, and PVSC models also performed well, but the NN did not. In fact, the performance of the NN was not much better than what could be achieved by randomly guessing the binary label with equal probability of choosing one or zero. The PVC model that performed best at identifying heart failure among patients is *gls-EF/complete/2*, and it attains an accuracy of 0.76, sensitivity of 0.81, specificity of 0.72 and DOR of 10.85.

In the patient diagnosis case study the PVSC model is regarded as the top performer. Here too, it was a close call between the PVSC, PVC and TSC models. The patient diagnosis dataset was skewed as there were 170 patients with a heart disease, and only 30 healthy patients. For this reason it is probable that the NN was unable to generalize the feature of the healthy patients, because almost all the variations of the NN ended up always making the prediction that the patient was diseased yielding a score of 0 in specificity. The PVSC model that performed best at predicting patient diagnosis is *gls-rls/KNN*, and it attains an accuracy of 0.93, sensitivity of 0.95, specificity of 0.82 and DOR of 84.53.

In the segment indication case study only the TSC and NN models were compared, and for a change of pace it was only the NN that was chosen as the best performer. The TSC model did not perform much worse, in fact it performed better than the NN in many respects. The key reason for why the NN was preferred was because it had a more balanced sensitivity, and specificity scores than the TSC model. The NN model that performed best at predicting segment indication is *downsampled*, and it attains an accuracy of 0.74, sensitivity of 0.74, specificity of 0.72 and DOR of 8.38.

Chapter 8

Discussion

In the results chapter, the performance results were presented in the order of the different target variables that were explored. In the discussion chapter a different approach is taken, and the each model will be discussed individually based on their performance in the case studies.

8.1 Time-series Clustering

Before dissimilarity was measured between strain curves, curves were preprocessed in one of four ways. Curves were either: not preprocessed, scaled between zero and one, normalized between zero and one or z-score normalized. The TSC model was implemented by using DTW distance between strain curves as a dissimilarity measure to achieve a shape-based TSC model. All the dissimilarity measures between a specific strain curve of one patient to the same strain curve of every other patient were combined into a dissimilarity matrix. If the dataset represented patients with more than one strain curve the dissimilarity matrices of each indivial strain curves were added together, such that there was a single dissimilarity matrix that represented the dissimilarity between the patients. The dissimilarity matrix was then passed to the hierarchical agglomorative clustering algorithm which started out with each patient as an indivial cluster, and merged clusters together based on a specific linkage criteria. Seven linkage criteria were tested: single, complete, average, ward, centroid, median and weighted. The clustering model was calculated at the different number of cluster centers between two and nine. The ARI was estimated for the all the cluster assignments generated, and the different target variables. For the cluster assignments yielded by a clustering model evaluated at two cluster centers the accuracy, sensitivity, specicity and DOR was also calculated.

The TSC models did not perform best in any of the case studies, but variations of the TSC models generally yielded results with high performance in terms of accuracy, sensitivity and specicity. In the heart failure case study the best variation of the TSC model achieved the highest sensitivity and DOR, but it was outperformed by the best variation of the PVC model overall. In the patient diagnosis case study the best variation of the TSC model outperformed the best variation of the PVC model, but they were both outperformed by the best PVSC model. In the segment indication case study the best variation of the TSC model attains the highest accuracy, specicity and DOR, but is outperformed by the NN because it attains a higher sensitivity score, and thereby attains a more balanced accuracy in the positives and negatives. As discussed in section REFERENCE, a challenge for all statistical models is the "curse of dimensionality". Briefly described, in ML, and data mining the curse of dimensionality refers to the issue of attaining a good balance between the number of dimensions that an object is represented in, and the number of objects used to train and/or evaluate the model. In the heart failure and patient diagnoses case studies the TSC models that perform best in terms of DOR,

and ARI are the models that use datasets where there are objects are represented by fewer dimensions. A reason for this could be that for 200 patients, the heart failure diagnoses, and patient diagnoses are most seperable for the TSC models when only one strain curve is used. The curve that then gives the easiest separation of patients is then the 2CH GLS curve. In the heart failure study the TSC models that attain the five best performing models in terms of DOR and ARI only use the GLS curve from the 2CH view, meaning that these methods only use one of 21 possible curves. This can be confirmed from table 7.1 and 7.2. In the patient diagnoses study one can see from table 7.8 that the five methods that attain the highest DOR also only use the GLS curve from the 2CH view. These two observations support the claim that at a dataset size of 200 objects using fewer strain curves makes it easier for TSC models to separate heart failure diagnoses, and patient diagnoses. An observation that does not directly support this claim is that in the patient diagnosis case study, the TSC models that attain the four highest ARI use a combination of GLS and RLS curves in the 4CH view, or use the GLS curves from all views. However, these methods also only use three and seven of 21 curves in total, so this observation does not negate the claim entirely. In all case studies it was found that TSC models that performed best in terms of DOR, and ARI used no preprocessing. In some cases models using scaling as a form of preprocessing yielded the same cluster assignments, which could indicate that scaling the curves before measuring dissimilarity does not make much of a difference. Since TSC models using normalization or z-score normalization as a form of preprocessing were not among the top five methods in terms of DOR, or ARI in any of the case studies the argument could be made that these form of preprocessing are not suited when using DTW as a dissimilarity on left ventrice strain curves. Of the seven linkages tested, it was the centroid, weighted and ward linkages that went into the TSC models that performed best at predicting heart failure, patient diagnosis and segment indication respectively, in the different case studies. However, the single, complete and average linkages also went into the methods that appeared in the top five candidates in terms of DOR, or ARI. So it is not possible to say certainly that all linkages other than centroid, weighted and ward linkages are not suited for clustering left ventricle strain curves, but one can say with some degree of certainty that the median linkage does not go into any of the TSC models that perform well in any of the three case studies. When calculating the dissimilarity matrix of a set of 200 curves, it took approximately 0.3 seconds using the C-optimized functions of the dtaidistance library. The time it took to compute the clustering varied between 0.15 and 0.45 seconds depending on what linkage was used. The single linkage criteria was found to be the fastest, and the complete linkage was found to be the slowest. That the single linkage was the fastest could is to be expected, as it fairly easy to compute. However, it was unexpected that the complete linkage was the one that took the longest time to compute as one would expect the more complex linkages such as the ward linkage to take the longest time to compute. When the size of the dataset was increased to approximately 3600 curves it took 162 seconds to compute the dissimilarity matrix. This increase in run time is in agreement with the time complexity of the DTW algorithm described in section REFERENCE. In addition, the time it took to compute the clustering after the dissimilarity matrix was computed also increased to vary between 3 seconds for the single linkage, and 871 seconds for the ward linkage. So for a bigger dataset the run time of the different linkages were more as expected. Although these run-times are attained with a with a regular desktop Lenove G510 laptop, it illustrates possible challenge of how run-time of the calculations of the dissimilarity matrix, and clustering increase quadratically with the size of the dataset. It was often found that the PVC models that used EF in addition to peak systolic strain values performed better than the PVC models that only used strain values. It would be interesting to see whether incorporating EF in the TSC model would improve its performance as well. Since the hierarchical agglomorative clustering algorithm is uses dissimilarity matrix to cluster objects, it should be fairly straight-forward to calculate the dissimilarity matrix between a patients EF values, and add that to the dissimilarity matrices of the indivial curves. One

could also consider the approach taken by CITATION, where they split the strain curves of one heart cycle into systolic, and diastolic strain curves, and pass them to the model separately. Although the authors achieved good results with this, they also say that annotating points of every strain curve as systolic or diastolic is very time consuming.

8.2 Peak-value Clustering

The PVC model was implemented in a similar fashion as the TSC model. The datapoints used to represent patients were passed to an implementation of hierarchical agglomorative clustering in scikit-learn. The dissimilarity between patients was measured as the Euclidean distance between the dimensions used to represent them. The scikit-learn implementation did not have all the same clustering linkages available as the scipy implementation used for TSC, so only the following four linkages were tested: single, complete, average and ward. The evaluation procedure for the PVC model was the same as the procedure used for TSC. The best variations of the PVC model had a high performance in the heart failure, and in the patient diagnosis case studies. It was chosen as the best model in the heart failure case study, but was closely followed by the TSC, and PVSC models. In the patient diagnosis case study the best variation of the PVC models attained the highest specicity, and second highest DOR of the three models compared. However, it was outperformed by both the TSC, and PVSC models due to its low sensitivity. In both case studies PVC models that used datasets that were a combination of peak systolic strain values and EF performed consistently better than the models than only used the strain values. This is to some degree expected in the heart failure case study, as EF is parameter that is established in the current medical procedures used to diagnose patients with heart failure. In the heart failure case study it was the complete linkage which was used in the model that was chosen as the best performer, but the ward, and average linkages were also used in the models that attained the top five DOR and ARI scores. In the patient diagnosis case study the complete linkages was also used in the model that was chosen as the best performer. In both case studies where PVC models where tested the models that were chosen as the best performers used the complete linkage, but the average and ward linkages were also used by other model variations that attained the five highest DOR and ARI. Hence, for PVC models using peak systolic strain values, and EF to identify heart failure among patients, and patient diagnosis the single linkage was not found to be suited. Since a scikit learn implementation was used for the PVC model, it was not possible to separate run-time of the dissimilarity calculation and the clustering itself. However, Euclidean distance is known to scale linearly with the number of dimensions per object, and number of objects in the dataset. Since the underlying algorithm used by scikit learn is the same as the one used by scipy it is assumed that it would perform similarily to the TSC model in terms of run time.

8.3 Neural Networks

For the NN two types of preprocessing were tested in addition the option of not preprocessing at all, upsampling the curves to the highest sample rate in the dataset and downsampling the curves to the lowest sample rate. The curves of the dataset were then passed as input to the NN architecture detailed in section REFERENCE together with the relevant target variables. The NN was trained for five epochs using SGD with back-propagation. To validate the NN models 10-fold cross validation was used, at the end of each fold the TP, TN, FP, and FN of the model were noted. After the NN had effectively attempted to predict every object of the dataset all the TP, TN, FP, and FN were summed and this grand total was used to estimate the models accuracy, sensitivity, specicity and DOR. The NN models performed worst of the four model groups in the heart failure case study, and the patient diagnosis case study. However, it attained the highest sensitivity in the segment indication case study, and was chosen as the best performing model

because its sensitivity and specicity were more balanced than the TSC model. In the patient diagnosis case study close to all of the NN models predicted all the patients to be unhealthy. It is evident that a NN with the architecture used in this assignment was not suited to classify patient diagnosies with a skewed dataset of only 170 unhealthy patients and 30 healthy patients. It is the authors opinion that the reason that the NN models performed so bad at predicting patient diagnosis is an aspect of "the curse of dimensionality", and that the network was not able to generalize the characteristics of healthy patients in the study, and therefore minimized loss function by predicting the outcome that was most probable. From table 10.6 one can see that the top nine variations of the NN model that performed best in the heart failure case study with regard to DOR, were models that used only the GLS curve from a single view, which supports the claim that Since the different NN models differed in architecture depending on how many curves were used to represent one patient, they also varied in the number of trainable parameters they have. The NN models which only take a single strain curve as input have 39457 trainable parameters, and the NN models that take 21 curves as input have 80417 trainable parameters. Even though there is no exact ratio of how big a dataset should be with regard to how many trainable parameters a model has, between 40 and 80 thousand parameters for a dataset of size 200 is likely too many trainable parameters. On the other hand, the NN model was chosen as the best performing model at predicting segment indication. In that case study though the size of the dataset is significantly larger, and each object is represented by a single curve. Considering that the architecture of the NN was given, and not developed specifically for this classification problem the performance that the model achieves is significant. It is the authors opinion that if more time is spent adapting the model to the dataset at hand, even better performances are within reach for the NN models. Especially for the segment indication classification problem, since it is much bigger than the two other datasets. There are alternatives to SGD that could be tested such as batch gradient descent, and the most popular choice mini-batch gradient descent which is a middle road between the two, and is often considered the best alternative REFERENCE. There is also the GRU cells that are an alternative to the LSTM cells. Like LSTM cells, GRU cells are able capture time-dependent connections. GRU cells are simpler than LSTM cells in composition, and are said to require less training data, to achieve the same accuracies as LSTM cells REFERENCE. The biggest flaw of the NN models is that they have so many trainable parameters compared to the number of objects in the datasets they are applied on. So the first improvements that could be done the NN models is reduce its complexity, by removing layers or reducing the number of filters and units in individual layers. One could also consider introducing layers that reduce complexity such as a max pooling layers, which for time series can be considered as a max-value filter where only the highest value in a segment of a curve is kept on. Dropout layers are also a technique that are used frequently when NN architecture become deep and complex, they introduce the probability that any particular perceptron in the layers preceding the dropout layer can "drop out" meaning that they become inactive. In complex NN architectures it is often found that during training the model becomes overly dependent on certain perceptrons, and specific paths through the network. This leads to the NN not entirely utilizing all the perceptrons at its disposal, and the accuracy suffers. It is found that by adding a probability that any given neuron can drop out during training remedies this effect, and can increase accuracy overall. Training, and validating the NN models were one of the more time-consuming computations required. The time it took to train the network depended on what dataset was used, which makes sense as increasing the number of curves the NN can take as input also increases the number of trainable parameters that need to be trained for each step of the SCG algorithm. When validating the NN models, a single fold in the 10-fold cross validation took approximately 100 seconds in the heart failure, and patient diagnosis case studies. The time it took to execute one fold in the segment indication case study took approximately 640 seconds (11 min) However, these times do not reflect the times it will take to use the NN to evaluate new cases after training, so the same challenge one has with

clustering is not as pressing should the aim be to deploy the NN in a real-time clinical setting.

8.4 Peak-value Supervised Classifiers

The different peak-value datasets are passed the different supervised classifiers in the model group. The different datasets are detailed in section 6.1, and the different supervised classifiers tested are detailed in section REFERENCE. Each combination of dataset and classifiers is validated by a 10-fold cross-validation in the same manner as the NN. In the heart failure case study the best PVSC model outperformed the best variations of the TSC, and NN models and had a performance that was on par with the PVC, although the best PVC model was ultimately deemed better in the end. In the patient diagnosis case study the best PVSC model attained the highest accuracy, sensitivity and DOR of the four model groups, and it was deemed the best model group at predict patient diagnosis. What should be adressed is the fact that the distribution of the DOR for the different PVSC models, differ from the DOR distributions of the other models in some key ways. In both the heart failure case study, and the patient diagnosis case study the distribution of DOR for variations of TSC, PVC and NN models are highly concentrated around zero. For the PVSC models the lowest DOR attained by a PVSC model in the heart failure study is 1.94, and the lowest DOR attained by a PVSC model in the patient diagnosis case study is 3.68. In the heart failure case study it is especially evident that the DOR of the different PVSC models is distributed differently than the DOR of the other models. It can be confirmed from figure 7.6 that the distribution of DOR for the PVSC is especially concentrated in the range between four to eight. The significance of this difference of DOR distribution is two-fold, the first thing to keep in mind is that not very much time was spent optimizing the hyperparameters of the PVSC models as it falls outside the scope of this thesis, and that in contrast to the clustering models the outcome of the PVSC model is probabilistic in the sense that it is highly dependent on the initial conditions of the model before it is trained. Since the DOR distribution of PVSC models in the heart failure, and patient diagnosis case studies are distributed higher in general than the TSC and PVC models, and that the PVSC are configured with what can be considered as "standard hyperparameters" it is probable that spending time on optimizing the hyperparameters of the PVSC models, and testing different initial conditions could improve the performance of all the PVSC models. The time it took to train and validate the PVSC models varied, and was highly dependent on the dimensions of the dataset and which specific ML model was used. The shortest training time encountered was at 201 seconds, and the longest was at 365 seconds. These were the shortest training times encountered among the four model groups. Similarily to the NN model the training times of the PVSC models do not hinder their ability to make predictions in real time, and deploy them in a clinical setting.

Chapter 9

Conclusion

The main objective of this thesis, as stated in section 1.2 have been explore whether a ML model using longitudinal strain values as input can identify whether a patient has heart failure, if a patient has a heart disease and if an individual segment in a patients left ventricle is acting abnormally. The main objective is divided into two sub-objectives that decided the direction and scope of the thesis: Which type of ML model will perform best, a supervised or unsupervised learning model, and what type of longitudinal strain data will yield the best performance for the ML models, longitudinal strain curves or peak systolic strain values in combination with EF.

A dataset of 200 patients was used to fulfill these objectives. The models that used combinations of GLS, and RLS curves from different views were a TSC model and an ANN, which were tested to classify heart failure among patients, patient diagnosis and whether individual left ventricle segments were acting abnormally. In addition to varying the dataset used with these models different forms of preprocessing was tested for both models, and different linkages were tested for the TSC model. The models that used peak systolic strain values were a PVC model, and a 11 different PVSC, they were only applied to identify heart failure among patients, and patient diagnosis. To assess the performance of the supervised models accuracy, sensitivity, specificity and DOR were used as evaluation metrics. To evaluate the unsupervised models the same metrics were used as for the supervised models, in addition to using the ARI to determine whether clustering models evaluated at a number of cluster centers greater than two could provide better performance than models evaluated at two cluster centers. When making a choice as to which model variation performed best within their respective model groups, and which model performed best overall the models were sorted in descending order of the DOR score they attained, the models which attained the highest DOR and accuracy while at the same time maintaining a balanced relationship of sensitivity and specificity were then chosen as the best performing models. For the clustering models, an additional evaluation was done with respect to ARI. If there were clustering models evaluated at a number of cluster centers greater than two that attained an ARI greater than the best performing two-cluster-center model, an attempt was made to visualize the result. Further, it was evaluated whether combining the clusters of the model with more than two centers could yield a better performance than the two-cluster-center model.

The overall consensus from the results are that it is possible to implement an ML model that uses longitudinal strain as input, and that can predict one of the three target varables. However, there was not a single model that performed best at predicting all the target variables. The model that performed best at identifying heart failure among patients was a variation of the PVC model which used a combination of peak systolic GLS values and EF as input data, used the

complete linkage and was evaluated at two cluster centers. This method attained an accuracy of 0.76, sensitivity of 0.81, specificity of 0.72 and DOR of 10.85. The model that performed best at predicting patient diagnosis was one of the PVSC models that used the KNN classifier trained on a combination of peak systolic GLS, and RLS values. It attained an accuracy of 0.93, a sensitivity of 0.95, a specificity of 0.82 and a DOR of 84.53. In the segment indication case study, the ANN that downsampled all the individual RLS curves to the lowest sample rate of all the curves was chosen as the best model. That model attained an accuracy of 0.74, sensitivity of 0.74, specificity of 0.75 and DOR of 8.38.

It was found that PVC, and PVSC models that used a combination of peak strain values and EF generally performed better at predicting heart failure than variations that used peak strain values alone. The ANN was not able to generalize the features of healthy patients in the patient diagnosis case study at all, and did not perform particularily well in the heart failure case study either. It is the authors opinion that this is because the architecture of the ANN is to complex to be trained solely on a dataset of 200 patients. This conclusion was drawn based on the fact that the ANN had between 40 and 80 thousand trainable depending on how many curves were used as input. This statement is also supported by the fact that the ANN performed significantly better, when applied to classify single curves on a dataset of size 3600 curves. The variations of TSC models that used no preprocessing performed better in general than the variations that used normalization, z-normalization or scaling, meaning that purely shape-based TSC is not optimal for clustering left ventricle strain curves for diagnosing patients.

9.1 Future Work

It is the authors opinion that there are two continuations of this work that show good promise. Since the scope of this thesis has been quite wide there has not been enough time to do a deep dive into any of the specific models, so both of the suggestions are deep dives into specific models since a broad comparison has now been made.

Development of an Artificial Neural Network for Segment Indication

Given that the ANN performed so well at identifying the binary segment indication, it is probable that by spending more time adapting the architecture to the segment indication dataset one could achieve performances that are better than the ones attained in this piece of work. One could start with the architecture used in this assignment, and attempt to reduce the complexity of the architecture by adding pooling layers, or dropout layers. It should be tested whether using GRU cells could could improve the accuracy of the ANN as they are known to require less data than LSTM cells to generalize the difference between different segment labels. One should also experiment with variations of SGD for training the network, such as batch GD and mini-batch GD. If concentrating mainly on an ANN solution one could also test if the resulting model is capable of dealing with segment indication when multiple classes are used.

Development of Peak-value Supervised classifiers

Recall that the PVSC models performed best at predicting patient diagnosis. As mentioned in section 8.4, although the PVSC did not perform best at identifying heart failure in patients, the distribution of the DOR for the PVSC models was shifted significantly higher, and centered higher than the DOR distribution of the TSC, PVC and ANN models. Since there was not time to optimize the hyperparameters of the individual classifiers in the PVSC group, this shift in distribution indicates that there is some lost potential as to what performance these models could attain. Therefore, it is probable that by spending more time on adapting the individual

classifiers to the heart failure, and patient diagnosis datasets one could produce models that yield higher scores in all evaluation metrics.

Chapter 10

Appendix

This is the appendix

10.1 Raw model results

10.1.1 Time-series Clustering

Table 10.1: Classification results of applying TSC to identify heart failure among patients. The results are sorted in descending order of DOR, although DOR is not included.

Dataset-Method	TP	TN	FP	FN
gls/2CH/regular/centroid/2	86	62	35	13
gls/2CH/scaled/centroid/2	86	62	35	13
gls/2CH/regular/average/2	84	63	34	15
gls/2CH/scaled/average/2	84	63	34	15
gls-rls/2CH/scaled/ward/2	81	65	32	18
rls/APLAX/scaled/weighted/2	90	45	52	9
gls-rls/APLAX/regular/median/2	26	93	4	73
gls-rls/APLAX/scaled/weighted/2	90	43	54	9
rls/APLAX/scaled/average/2	82	60	37	17
gls/2CH/regular/ward/2	80	63	34	19
gls/2CH/scaled/ward/2	80	63	34	19
gls-rls/2CH/scaled/complete/2	74	70	27	25
gls-rls/4CH/regular/weighted/2	98	7	90	1
rls/2CH/scaled/ward/2	77	66	31	22
gls/2CH/regular/complete/2	75	68	29	24
gls/2CH/scaled/complete/2	75	68	29	24
gls/4CH/scaled/centroid/2	77	64	33	22
gls/4CH/regular/centroid/2	77	64	33	22
rls/all-views/regular/complete/2	86	49	48	13
gls/all-views/regular/weighted/2	88	44	53	11
gls/all-views/regular/centroid/2	74	67	30	25
gls-rls/2CH/regular/complete/2	73	68	29	26
gls-rls/2CH/regular/ward/2	78	62	35	21
gls-rls/APLAX/scaled/average/2	81	57	40	18
gls/all-views/scaled/average/2	73	67	30	26
gls/all-views/regular/median/2	21	93	4	78
gls-rls/4CH/regular/complete/2	91	34	63	8

gls/4CH/regular/complete/2	74	64	33	25
gls/4CH/scaled/complete/2	74	64	33	25
gls/all-views/scaled/ward/2	67	71	26	32
gls/all-views/regular/ward/2	67	71	26	32
gls-rls/4CH/scaled/weighted/2	85	47	50	14
gls/all-views/scaled/complete/2	66	71	26	33
rls/2CH/regular/complete/2	67	70	27	32
gls/all-views/regular/average/2	62	74	23	37
gls/all-views/regular/complete/2	62	74	23	37
rls/all-views/scaled/ward/2	59	76	21	40
gls-rls/4CH/scaled/average/2	60	75	22	39
gls-rls/all-views/regular/complete/2	60	75	22	39
gls-rls/all-views/scaled/weighted/2	60	75	22	39
gls/APLAX/regular/ward/2	65	71	26	34
gls/APLAX/regular/median/2	65	71	26	34
gls-rls/all-views/regular/ward/2	61	74	23	38
rls/all-views/scaled/weighted/2	58	76	21	41
gls-rls/APLAX/scaled/centroid/2	58	76	21	41
gls-rls/all-views/regular/centroid/2	62	73	24	37
gls/APLAX/regular/average/2	63	72	25	36
rls/APLAX/scaled/ward/2	59	75	22	40
rls/all-views/scaled/complete/2	59	75	22	40
gls-rls/APLAX/scaled/complete/2	41	85	12	58
gls/APLAX/regular/centroid/2	65	70	27	34
gls-rls/all-views/scaled/average/2	60	74	23	39
gls/APLAX/regular/complete/2	47	82	15	52
gls/all-views/scaled/centroid/2	61	73	24	38
gls-rls/all-views/scaled/centroid/2	61	73	24	38
gls/4CH/regular/median/2	24	91	6	75
gls/4CH/regular/weighted/2	24	91	6	75
gls/4CH/scaled/weighted/2	24	91	6	75
gls/4CH/scaled/median/2	24	91	6	75
gls-rls/APLAX/regular/average/2	58	75	22	41
rls/APLAX/regular/ward/2	58	75	22	41
gls-rls/all-views/regular/average/2	58	75	22	41
rls/APLAX/regular/weighted/2	46	82	15	53
gls/all-views/scaled/weighted/2	13	94	3	86
gls-rls/4CH/scaled/ward/2	56	76	21	43
rls/4CH/scaled/average/2	56	76	21	43
rls/all-views/scaled/average/2	54	77	20	45
gls-rls/APLAX/scaled/ward/2	54	77	20	45
rls/all-views/regular/average/2	54	77	20	45
gls-rls/all-views/scaled/complete/2	54	77	20	45
gls-rls/4CH/scaled/complete/2	54	77	20	45
rls/APLAX/scaled/complete/2	58	74	23	41
gls-rls/APLAX/regular/ward/2	55	76	21	44
gls-rls/all-views/scaled/ward/2	55	76	21	44
rls/4CH/regular/complete/2	64	69	28	35
gls/APLAX/regular/weighted/2	36	86	11	63
gls/2CH/scaled/median/2	57	74	23	42
gls/2CH/scaled/weighted/2	57	74	23	42

gls/2CH/regular/weighted/2	57	74	23	42
gls/2CH/regular/median/2	57	74	23	42
gls-rls/APLAX/regular/centroid/2	51	78	19	48
gls-rls/4CH/regular/ward/2	54	76	21	45
gls-rls/4CH/regular/average/2	54	76	21	45
rls/APLAX/regular/complete/2	54	76	21	45
rls/4CH/scaled/ward/2	52	77	20	47
rls/2CH/normalized/median/2	30	88	9	69
rls/all-views/normalized/weighted/2	98	4	93	1
rls/APLAX/normalized/median/2	98	4	93	1
rls/2CH/scaled/complete/2	60	71	26	39
gls/4CH/scaled/ward/2	43	82	15	56
gls/4CH/regular/ward/2	43	82	15	56
gls-rls/APLAX/regular/complete/2	73	58	39	26
rls/all-views/regular/ward/2	54	75	22	45
gls-rls/all-views/regular/weighted/2	46	80	17	53
gls/all-views/scaled/median/2	65	65	32	34
rls/4CH/scaled/complete/2	58	71	26	41
rls/4CH/regular/ward/2	52	75	22	47
rls/2CH/regular/ward/2	45	79	18	54
gls-rls/APLAX/regular/weighted/2	29	86	11	70
rls/4CH/regular/weighted/2	97	6	91	2
gls-rls/all-views/normalized/ward/2	27	85	12	72
rls/all-views/normalized/complete/2	35	80	17	64
gls-rls/4CH/normalized/ward/2	53	65	32	46
gls-rls/4CH/z-normalized/ward/2	60	58	39	39
gls-rls/2CH/z-normalized/ward/2	78	36	61	21
gls-rls/4CH/scaled/median/2	96	6	91	3
rls/4CH/z-normalized/ward/2	60	56	41	39
rls/2CH/z-normalized/weighted/2	98	2	95	1
rls/all-views/scaled/median/2	98	2	95	1
gls-rls/2CH/z-normalized/complete/2	98	2	95	1
rls/all-views/normalized/ward/2	58	56	41	41
gls/2CH/z-normalized/ward/2	70	42	55	29
rls/all-views/z-normalized/ward/2	50	61	36	49
gls-rls/APLAX/normalized/ward/2	25	81	16	74
gls/APLAX/normalized/complete/2	22	83	14	77
gls-rls/APLAX/normalized/complete/2	23	82	15	76
gls-rls/all-views/z-normalized/ward/2	51	59	38	48
rls/APLAX/normalized/ward/2	24	81	16	75
gls/all-views/z-normalized/ward/2	54	55	42	45
gls/4CH/z-normalized/complete/2	47	61	36	52
gls-rls/all-views/z-normalized/complete/2	49	59	38	50
rls/2CH/normalized/ward/2	74	32	65	25
gls/4CH/normalized/ward/2	39	67	30	60
rls/4CH/normalized/complete/2	77	28	69	22
rls/2CH/normalized/complete/2	77	28	69	22
gls/APLAX/z-normalized/complete/2	40	65	32	59
gls-rls/APLAX/z-normalized/complete/2	42	63	34	57
gls-rls/APLAX/z-normalized/ward/2	43	62	35	56
rls/all-views/z-normalized/complete/2	48	57	40	51

gls/all-views/normalized/ward/2	37	67	30	62
gls-rls/2CH/normalized/ward/2	65	39	58	34
rls/2CH/z-normalized/ward/2	49	55	42	50
gls/APLAX/z-normalized/ward/2	37	66	31	62
gls-rls/all-views/normalized/complete/2	15	85	12	84
gls-rls/2CH/normalized/complete/2	70	33	64	29
rls/4CH/normalized/ward/2	79	23	74	20
gls/2CH/normalized/ward/2	62	41	56	37
rls/APLAX/normalized/complete/2	34	68	29	65
gls-rls/4CH/normalized/complete/2	35	67	30	64
rls/APLAX/z-normalized/complete/2	60	42	55	39
rls/APLAX/z-normalized/ward/2	30	70	27	69
gls/4CH/z-normalized/ward/2	33	67	30	66
gls-rls/APLAX/normalized/weighted/2	78	22	75	21
gls/APLAX/normalized/ward/2	76	24	73	23
gls/all-views/z-normalized/complete/2	64	35	62	35
gls/all-views/normalized/complete/2	84	15	82	15
rls/all-views/regular/median/2	94	5	92	5
gls-rls/2CH/z-normalized/weighted/2	97	2	95	2
rls/4CH/scaled/median/2	98	1	96	1
rls/4CH/regular/average/2	98	1	96	1
gls-rls/4CH/regular/single/2	98	0	97	1
gls-rls/all-views/scaled/median/2	98	0	97	1
gls-rls/all-views/z-normalized/centroid/2	98	0	97	1
gls-rls/APLAX/z-normalized/weighted/2	98	0	97	1
gls/all-views/normalized/single/2	98	0	97	1
gls-rls/APLAX/normalized/centroid/2	98	0	97	1
gls-rls/APLAX/normalized/average/2	98	0	97	1
gls-rls/all-views/scaled/single/2	98	0	97	1
gls-rls/APLAX/z-normalized/single/2	98	0	97	1
gls-rls/APLAX/normalized/median/2	98	0	97	1
gls-rls/APLAX/normalized/single/2	98	0	97	1
gls/all-views/z-normalized/single/2	98	0	97	1
gls-rls/APLAX/z-normalized/centroid/2	98	0	97	1
gls-rls/2CH/normalized/centroid/2	98	0	97	1
gls-rls/4CH/normalized/single/2	98	0	97	1
gls-rls/2CH/z-normalized/centroid/2	98	0	97	1
gls-rls/2CH/normalized/average/2	98	0	97	1
gls-rls/2CH/normalized/median/2	98	0	97	1
gls-rls/2CH/z-normalized/single/2	98	0	97	1
gls-rls/2CH/normalized/single/2	98	0	97	1
gls-rls/2CH/z-normalized/average/2	98	0	97	1
gls-rls/2CH/regular/weighted/2	98	0	97	1
gls-rls/2CH/regular/median/2	98	0	97	1
gls-rls/2CH/regular/centroid/2	98	0	97	1
gls/all-views/normalized/centroid/2	98	0	97	1
gls-rls/2CH/regular/average/2	98	0	97	1
gls/all-views/normalized/median/2	98	0	97	1
gls-rls/2CH/regular/single/2	98	0	97	1
gls/all-views/normalized/weighted/2	98	0	97	1
gls-rls/2CH/scaled/single/2	98	0	97	1

gls-rls/4CH/normalized/average/2	98	0	97	1
gls-rls/4CH/scaled/single/2	98	0	97	1
gls-rls/4CH/z-normalized/weighted/2	98	0	97	1
gls-rls/4CH/z-normalized/median/2	98	0	97	1
gls-rls/4CH/z-normalized/centroid/2	98	0	97	1
gls-rls/2CH/scaled/average/2	98	0	97	1
gls/all-views/normalized/average/2	98	0	97	1
gls-rls/4CH/z-normalized/average/2	98	0	97	1
gls-rls/4CH/z-normalized/complete/2	98	0	97	1
gls-rls/4CH/z-normalized/single/2	98	0	97	1
gls-rls/2CH/scaled/centroid/2	98	0	97	1
gls-rls/4CH/normalized/median/2	98	0	97	1
gls-rls/4CH/normalized/centroid/2	98	0	97	1
gls-rls/2CH/scaled/weighted/2	98	0	97	1
gls-rls/4CH/normalized/weighted/2	98	0	97	1
gls/4CH/normalized/median/2	98	0	97	1
gls-rls/all-views/z-normalized/single/2	98	0	97	1
gls/2CH/z-normalized/median/2	98	0	97	1
gls/APLAX/normalized/single/2	98	0	97	1
gls/APLAX/normalized/average/2	98	0	97	1
gls/APLAX/normalized/centroid/2	98	0	97	1
gls/APLAX/normalized/median/2	98	0	97	1
gls/APLAX/normalized/weighted/2	98	0	97	1
gls/APLAX/z-normalized/single/2	98	0	97	1
gls/APLAX/z-normalized/average/2	98	0	97	1
gls/APLAX/z-normalized/centroid/2	98	0	97	1
rls/all-views/regular/single/2	98	0	97	1
rls/all-views/regular/weighted/2	98	0	97	1
rls/all-views/normalized/single/2	98	0	97	1
rls/all-views/normalized/centroid/2	98	0	97	1
rls/all-views/normalized/median/2	98	0	97	1
rls/all-views/z-normalized/single/2	98	0	97	1
rls/all-views/z-normalized/centroid/2	98	0	97	1
rls/all-views/z-normalized/median/2	98	0	97	1
rls/all-views/scaled/single/2	98	0	97	1
gls/2CH/z-normalized/weighted/2	98	0	97	1
gls/2CH/z-normalized/centroid/2	98	0	97	1
rls/4CH/regular/median/2	98	0	97	1
gls/2CH/z-normalized/average/2	98	0	97	1
gls/4CH/z-normalized/single/2	98	0	97	1
gls/4CH/z-normalized/average/2	98	0	97	1
gls/4CH/z-normalized/centroid/2	98	0	97	1
gls/4CH/z-normalized/median/2	98	0	97	1
gls/4CH/z-normalized/weighted/2	98	0	97	1
gls/4CH/normalized/centroid/2	98	0	97	1
gls/4CH/normalized/average/2	98	0	97	1
gls/4CH/normalized/complete/2	98	0	97	1
gls/4CH/normalized/single/2	98	0	97	1
gls/2CH/normalized/single/2	98	0	97	1
gls/2CH/normalized/complete/2	98	0	97	1
gls/2CH/normalized/average/2	98	0	97	1

gls/2CH/normalized/centroid/2	98	0	97	1
gls/2CH/normalized/median/2	98	0	97	1
gls/2CH/normalized/weighted/2	98	0	97	1
gls/2CH/z-normalized/single/2	98	0	97	1
gls/2CH/z-normalized/complete/2	98	0	97	1
rls/4CH/regular/single/2	98	0	97	1
rls/4CH/normalized/single/2	98	0	97	1
gls-rls/all-views/normalized/centroid/2	98	0	97	1
rls/2CH/z-normalized/single/2	98	0	97	1
gls/4CH/normalized/weighted/2	98	0	97	1
rls/2CH/scaled/single/2	98	0	97	1
rls/2CH/scaled/average/2	98	0	97	1
gls/all-views/z-normalized/centroid/2	98	0	97	1
rls/2CH/scaled/centroid/2	98	0	97	1
rls/2CH/scaled/median/2	98	0	97	1
rls/2CH/scaled/weighted/2	98	0	97	1
rls/APLAX/normalized/single/2	98	0	97	1
rls/APLAX/normalized/centroid/2	98	0	97	1
rls/APLAX/normalized/weighted/2	98	0	97	1
rls/APLAX/z-normalized/single/2	98	0	97	1
rls/APLAX/z-normalized/centroid/2	98	0	97	1
rls/APLAX/z-normalized/median/2	98	0	97	1
gls/all-views/z-normalized/average/2	98	0	97	1
gls-rls/all-views/regular/single/2	98	0	97	1
gls-rls/all-views/regular/median/2	98	0	97	1
gls-rls/all-views/normalized/single/2	98	0	97	1
rls/2CH/z-normalized/average/2	98	0	97	1
rls/2CH/normalized/centroid/2	98	0	97	1
rls/4CH/normalized/average/2	98	0	97	1
rls/2CH/normalized/average/2	98	0	97	1
rls/4CH/normalized/centroid/2	98	0	97	1
rls/4CH/normalized/median/2	98	0	97	1
rls/4CH/normalized/weighted/2	98	0	97	1
rls/4CH/z-normalized/single/2	98	0	97	1
rls/4CH/z-normalized/complete/2	98	0	97	1
rls/4CH/z-normalized/average/2	98	0	97	1
rls/4CH/z-normalized/centroid/2	98	0	97	1
rls/4CH/z-normalized/median/2	98	0	97	1
rls/4CH/z-normalized/weighted/2	98	0	97	1
rls/4CH/scaled/single/2	98	0	97	1
gls/all-views/z-normalized/weighted/2	98	0	97	1
rls/2CH/regular/single/2	98	0	97	1
gls/all-views/z-normalized/median/2	98	0	97	1
rls/2CH/regular/average/2	98	0	97	1
rls/2CH/regular/centroid/2	98	0	97	1
rls/2CH/regular/weighted/2	98	0	97	1
rls/2CH/normalized/single/2	98	0	97	1
rls/2CH/z-normalized/centroid/2	98	0	97	1
gls/all-views/regular/single/2	99	1	96	0
gls/all-views/scaled/single/2	99	1	96	0
gls/4CH/regular/single/2	99	1	96	0

gls/4CH/regular/average/2	99	1	96	0
gls/4CH/scaled/single/2	99	1	96	0
gls/4CH/scaled/average/2	99	1	96	0
gls/2CH/regular/single/2	99	1	96	0
gls/2CH/scaled/single/2	99	1	96	0
gls/APLAX/regular/single/2	99	1	96	0
rls/all-views/regular/centroid/2	99	0	97	0
rls/all-views/normalized/average/2	99	1	96	0
rls/all-views/z-normalized/average/2	2	97	0	97
rls/all-views/z-normalized/weighted/2	2	97	0	97
rls/all-views/scaled/centroid/2	99	0	97	0
rls/4CH/regular/centroid/2	99	1	96	0
rls/4CH/scaled/centroid/2	99	1	96	0
rls/4CH/scaled/weighted/2	99	1	96	0
rls/2CH/regular/median/2	99	0	97	0
rls/2CH/normalized/weighted/2	99	1	96	0
rls/2CH/z-normalized/complete/2	3	97	0	96
rls/2CH/z-normalized/median/2	99	2	95	0
rls/APLAX/regular/single/2	99	1	96	0
rls/APLAX/regular/average/2	99	1	96	0
rls/APLAX/regular/centroid/2	99	0	97	0
rls/APLAX/regular/median/2	99	1	96	0
rls/APLAX/normalized/average/2	99	2	95	0
rls/APLAX/z-normalized/average/2	2	97	0	97
rls/APLAX/z-normalized/weighted/2	99	1	96	0
rls/APLAX/scaled/single/2	99	1	96	0
rls/APLAX/scaled/centroid/2	99	0	97	0
rls/APLAX/scaled/median/2	99	1	96	0
gls-rls/all-views/normalized/average/2	2	97	0	97
gls-rls/all-views/normalized/median/2	99	1	96	0
gls-rls/all-views/normalized/weighted/2	99	1	96	0
gls-rls/all-views/z-normalized/average/2	2	97	0	97
gls-rls/all-views/z-normalized/median/2	99	0	97	0
gls-rls/all-views/z-normalized/weighted/2	2	97	0	97
gls-rls/4CH/regular/centroid/2	99	1	96	0
gls-rls/4CH/regular/median/2	99	1	96	0
gls-rls/4CH/scaled/centroid/2	99	1	96	0
gls-rls/2CH/normalized/weighted/2	99	1	96	0
gls-rls/2CH/z-normalized/median/2	99	2	95	0
gls-rls/2CH/scaled/median/2	99	0	97	0
gls-rls/APLAX/regular/single/2	99	1	96	0
gls-rls/APLAX/z-normalized/average/2	2	97	0	97
gls-rls/APLAX/z-normalized/median/2	99	0	97	0
gls-rls/APLAX/scaled/single/2	99	1	96	0
gls-rls/APLAX/scaled/median/2	99	1	96	0

Table 10.2: Classification results of applying TSC to identify patient diagnoses. The results are sorted in descending order of DOR, although DOR is not included.

Dataset-Method	TP	TN	FP	FN
gls/2CH/regular/centroid/2	119	27	2	48

gls/2CH/scaled/centroid/2	119	27	2	48
gls/2CH/scaled/average/2	116	27	2	51
gls/2CH/regular/average/2	116	27	2	51
gls/2CH/scaled/ward/2	112	27	2	55
gls/2CH/regular/ward/2	112	27	2	55
gls-rls/2CH/scaled/ward/2	111	27	2	56
gls-rls/2CH/regular/ward/2	111	27	2	56
rls/all-views/normalized/weighted/2	166	4	25	1
rls/2CH/scaled/ward/2	106	27	2	61
rls/all-views/regular/complete/2	130	25	4	37
rls/4CH/regular/weighted/2	165	6	23	2
gls/all-views/regular/centroid/2	102	27	2	65
gls/2CH/scaled/complete/2	102	27	2	65
gls/2CH/regular/complete/2	102	27	2	65
gls/all-views/regular/weighted/2	136	24	5	31
gls/all-views/scaled/average/2	101	27	2	66
gls-rls/2CH/regular/complete/2	100	27	2	67
rls/APLAX/scaled/average/2	116	26	3	51
gls-rls/2CH/scaled/complete/2	99	27	2	68
gls-rls/4CH/scaled/weighted/2	130	24	5	37
rls/2CH/regular/complete/2	92	27	2	75
gls/all-views/scaled/ward/2	91	27	2	76
gls/all-views/regular/ward/2	91	27	2	76
gls/all-views/scaled/complete/2	90	27	2	77
gls/APLAX/regular/centroid/2	90	27	2	77
gls/4CH/scaled/centroid/2	107	26	3	60
gls/4CH/regular/centroid/2	107	26	3	60
gls/APLAX/regular/median/2	89	27	2	78
gls/APLAX/regular/ward/2	89	27	2	78
gls-rls/4CH/regular/complete/2	145	20	9	22
gls-rls/APLAX/scaled/average/2	117	25	4	50
gls/APLAX/regular/average/2	86	27	2	81
gls/4CH/regular/complete/2	104	26	3	63
gls/4CH/scaled/complete/2	104	26	3	63
gls-rls/4CH/scaled/median/2	164	6	23	3
gls-rls/all-views/regular/centroid/2	84	27	2	83
rls/2CH/scaled/complete/2	84	27	2	83
gls/all-views/scaled/centroid/2	83	27	2	84
gls-rls/all-views/scaled/centroid/2	83	27	2	84
gls/all-views/regular/complete/2	83	27	2	84
gls/all-views/regular/average/2	83	27	2	84
rls/APLAX/scaled/weighted/2	135	22	7	32
gls-rls/all-views/regular/ward/2	82	27	2	85
gls-rls/all-views/scaled/average/2	81	27	2	86
gls-rls/all-views/scaled/weighted/2	80	27	2	87
gls-rls/all-views/regular/complete/2	80	27	2	87
gls-rls/4CH/scaled/average/2	80	27	2	87
gls-rls/2CH/z-normalized/complete/2	166	2	27	1
rls/2CH/z-normalized/weighted/2	166	2	27	1
rls/APLAX/scaled/ward/2	79	27	2	88
rls/all-views/scaled/complete/2	79	27	2	88

rls/APLAX/scaled/complete/2	79	27	2	88
gls/2CH/scaled/weighted/2	78	27	2	89
gls-rls/all-views/regular/average/2	78	27	2	89
gls/2CH/scaled/median/2	78	27	2	89
gls-rls/APLAX/regular/average/2	78	27	2	89
rls/all-views/scaled/ward/2	78	27	2	89
gls/2CH/regular/median/2	78	27	2	89
gls/2CH/regular/weighted/2	78	27	2	89
rls/APLAX/regular/ward/2	78	27	2	89
rls/all-views/scaled/weighted/2	77	27	2	90
gls-rls/APLAX/scaled/centroid/2	77	27	2	90
gls-rls/APLAX/scaled/weighted/2	136	21	8	31
gls-rls/4CH/regular/weighted/2	164	5	24	3
rls/4CH/scaled/average/2	75	27	2	92
gls-rls/4CH/scaled/ward/2	75	27	2	92
rls/all-views/regular/ward/2	74	27	2	93
gls-rls/APLAX/regular/ward/2	74	27	2	93
gls-rls/all-views/scaled/ward/2	74	27	2	93
gls-rls/4CH/regular/ward/2	73	27	2	94
gls-rls/4CH/regular/average/2	73	27	2	94
rls/APLAX/regular/complete/2	73	27	2	94
gls-rls/all-views/scaled/complete/2	72	27	2	95
rls/4CH/regular/ward/2	72	27	2	95
rls/all-views/regular/average/2	72	27	2	95
gls-rls/APLAX/scaled/ward/2	72	27	2	95
rls/all-views/scaled/average/2	72	27	2	95
gls-rls/4CH/scaled/complete/2	72	27	2	95
rls/4CH/regular/complete/2	89	26	3	78
gls-rls/APLAX/regular/complete/2	107	24	5	60
rls/4CH/scaled/complete/2	81	26	3	86
gls/all-views/scaled/median/2	93	25	4	74
gls-rls/2CH/z-normalized/weighted/2	165	2	27	2
rls/4CH/regular/average/2	166	1	28	1
rls/4CH/scaled/median/2	166	1	28	1
gls/APLAX/normalized/ward/2	134	14	15	33
rls/2CH/normalized/ward/2	126	16	13	41
rls/4CH/normalized/ward/2	137	13	16	30
rls/2CH/normalized/complete/2	131	14	15	36
gls-rls/APLAX/normalized/weighted/2	136	12	17	31
rls/4CH/normalized/complete/2	130	13	16	37
gls-rls/2CH/normalized/ward/2	111	17	12	56
gls-rls/2CH/normalized/complete/2	120	15	14	47
rls/APLAX/z-normalized/ward/2	124	14	15	43
gls/all-views/z-normalized/complete/2	113	16	13	54
gls-rls/4CH/normalized/complete/2	116	14	15	51
gls/all-views/normalized/ward/2	114	14	15	53
gls/all-views/normalized/complete/2	144	7	22	23
gls/APLAX/z-normalized/ward/2	113	14	15	54
gls/4CH/z-normalized/ward/2	117	13	16	50
gls/APLAX/z-normalized/complete/2	109	14	15	58
gls/4CH/normalized/ward/2	111	13	16	56

rls/2CH/z-normalized/ward/2	92	16	13	75
gls-rls/APLAX/z-normalized/ward/2	103	14	15	64
gls-rls/all-views/z-normalized/ward/2	93	15	14	74
gls-rls/2CH/z-normalized/ward/2	120	10	19	47
gls/all-views/z-normalized/ward/2	87	16	13	80
gls/4CH/z-normalized/complete/2	98	14	15	69
rls/all-views/z-normalized/ward/2	95	14	15	72
rls/APLAX/normalized/complete/2	114	10	19	53
gls-rls/APLAX/normalized/complete/2	135	6	23	32
gls-rls/4CH/normalized/ward/2	95	13	16	72
gls-rls/4CH/z-normalized/ward/2	83	15	14	84
rls/all-views/normalized/ward/2	83	15	14	84
rls/all-views/z-normalized/complete/2	92	13	16	75
rls/4CH/z-normalized/ward/2	86	14	15	81
gls-rls/APLAX/normalized/ward/2	132	6	23	35
gls/APLAX/normalized/complete/2	136	5	24	31
rls/APLAX/z-normalized/complete/2	97	11	18	70
gls/all-views/scaled/weighted/2	153	2	27	14
rls/APLAX/normalized/ward/2	132	5	24	35
gls/2CH/z-normalized/ward/2	105	9	20	62
gls-rls/APLAX/z-normalized/complete/2	100	9	20	67
rls/all-views/regular/median/2	158	1	28	9
gls-rls/all-views/z-normalized/complete/2	90	10	19	77
rls/all-views/normalized/complete/2	120	5	24	47
gls/2CH/normalized/ward/2	97	8	21	70
gls/all-views/regular/median/2	144	2	27	23
gls-rls/all-views/normalized/complete/2	142	2	27	25
gls/4CH/scaled/median/2	139	2	27	28
gls/4CH/scaled/weighted/2	139	2	27	28
gls/4CH/regular/weighted/2	139	2	27	28
gls/4CH/regular/median/2	139	2	27	28
gls/APLAX/regular/weighted/2	122	2	27	45
gls-rls/APLAX/regular/median/2	138	1	28	29
gls-rls/APLAX/scaled/complete/2	116	2	27	51
gls/4CH/scaled/ward/2	111	2	27	56
gls/4CH/regular/ward/2	111	2	27	56
rls/APLAX/regular/weighted/2	108	2	27	59
gls/APLAX/regular/complete/2	107	2	27	60
gls-rls/all-views/regular/weighted/2	106	2	27	61
rls/2CH/regular/ward/2	106	2	27	61
gls-rls/APLAX/regular/weighted/2	128	1	28	39
gls-rls/APLAX/regular/centroid/2	99	2	27	68
rls/4CH/scaled/ward/2	97	2	27	70
gls/4CH/z-normalized/weighted/2	166	0	29	1
gls-rls/4CH/scaled/single/2	166	0	29	1
gls/2CH/normalized/median/2	166	0	29	1
gls/2CH/normalized/centroid/2	166	0	29	1
gls/2CH/normalized/average/2	166	0	29	1
gls/2CH/normalized/complete/2	166	0	29	1
gls/2CH/normalized/single/2	166	0	29	1
gls-rls/2CH/regular/single/2	166	0	29	1

rls/4CH/scaled/single/2	166	0	29	1
gls-rls/4CH/z-normalized/weighted/2	166	0	29	1
gls/4CH/z-normalized/median/2	166	0	29	1
gls-rls/2CH/regular/centroid/2	166	0	29	1
gls-rls/2CH/regular/median/2	166	0	29	1
gls-rls/2CH/regular/weighted/2	166	0	29	1
gls-rls/2CH/normalized/single/2	166	0	29	1
gls/4CH/z-normalized/centroid/2	166	0	29	1
gls-rls/2CH/normalized/average/2	166	0	29	1
gls-rls/2CH/regular/average/2	166	0	29	1
gls-rls/4CH/z-normalized/median/2	166	0	29	1
gls-rls/2CH/normalized/centroid/2	166	0	29	1
gls-rls/4CH/normalized/average/2	166	0	29	1
gls-rls/4CH/regular/single/2	166	0	29	1
gls/2CH/z-normalized/weighted/2	166	0	29	1
gls/2CH/z-normalized/median/2	166	0	29	1
gls/2CH/z-normalized/centroid/2	166	0	29	1
gls/2CH/z-normalized/average/2	166	0	29	1
gls-rls/4CH/normalized/single/2	166	0	29	1
gls/2CH/z-normalized/complete/2	166	0	29	1
gls/2CH/z-normalized/single/2	166	0	29	1
gls-rls/4CH/z-normalized/centroid/2	166	0	29	1
gls-rls/4CH/normalized/centroid/2	166	0	29	1
gls-rls/4CH/normalized/median/2	166	0	29	1
gls-rls/4CH/normalized/weighted/2	166	0	29	1
gls-rls/4CH/z-normalized/single/2	166	0	29	1
gls-rls/4CH/z-normalized/complete/2	166	0	29	1
gls-rls/4CH/z-normalized/average/2	166	0	29	1
gls/2CH/normalized/weighted/2	166	0	29	1
gls/4CH/z-normalized/average/2	166	0	29	1
gls-rls/2CH/z-normalized/single/2	166	0	29	1
gls-rls/2CH/normalized/median/2	166	0	29	1
gls/all-views/normalized/weighted/2	166	0	29	1
gls/all-views/z-normalized/centroid/2	166	0	29	1
gls-rls/APLAX/normalized/centroid/2	166	0	29	1
gls-rls/APLAX/normalized/median/2	166	0	29	1
gls/all-views/z-normalized/average/2	166	0	29	1
gls-rls/APLAX/z-normalized/single/2	166	0	29	1
gls/all-views/z-normalized/single/2	166	0	29	1
gls-rls/APLAX/z-normalized/average/2	165	0	29	2
gls-rls/APLAX/z-normalized/centroid/2	166	0	29	1
gls-rls/all-views/scaled/median/2	166	0	29	1
gls-rls/APLAX/z-normalized/weighted/2	166	0	29	1
gls-rls/APLAX/scaled/single/2	166	0	29	1
gls/all-views/normalized/median/2	166	0	29	1
gls/all-views/normalized/centroid/2	166	0	29	1
gls/all-views/normalized/average/2	166	0	29	1
gls/all-views/normalized/single/2	166	0	29	1
gls-rls/APLAX/scaled/median/2	166	0	29	1
gls-rls/APLAX/normalized/average/2	166	0	29	1
gls/all-views/z-normalized/median/2	166	0	29	1

gls-rls/APLAX/normalized/single/2	166	0	29	1
gls/all-views/z-normalized/weighted/2	166	0	29	1
gls/4CH/z-normalized/single/2	166	0	29	1
gls-rls/2CH/z-normalized/average/2	166	0	29	1
gls/4CH/normalized/weighted/2	166	0	29	1
gls-rls/2CH/z-normalized/centroid/2	166	0	29	1
gls/4CH/normalized/median/2	166	0	29	1
gls-rls/2CH/scaled/single/2	166	0	29	1
gls/4CH/normalized/centroid/2	166	0	29	1
gls-rls/2CH/scaled/average/2	166	0	29	1
gls/4CH/normalized/average/2	166	0	29	1
gls-rls/2CH/scaled/centroid/2	166	0	29	1
gls-rls/2CH/scaled/weighted/2	166	0	29	1
gls-rls/APLAX/regular/single/2	166	0	29	1
gls/4CH/normalized/complete/2	166	0	29	1
gls/4CH/normalized/single/2	166	0	29	1
gls/all-views/scaled/single/2	166	0	29	1
gls/APLAX/regular/single/2	166	0	29	1
gls/APLAX/normalized/single/2	166	0	29	1
rls/4CH/z-normalized/weighted/2	166	0	29	1
rls/APLAX/regular/single/2	166	0	29	1
rls/2CH/scaled/single/2	166	0	29	1
rls/4CH/normalized/average/2	166	0	29	1
rls/2CH/scaled/average/2	166	0	29	1
rls/4CH/normalized/single/2	166	0	29	1
rls/2CH/scaled/centroid/2	166	0	29	1
rls/2CH/scaled/median/2	166	0	29	1
rls/2CH/scaled/weighted/2	166	0	29	1
rls/4CH/regular/median/2	166	0	29	1
rls/2CH/z-normalized/centroid/2	166	0	29	1
rls/APLAX/regular/average/2	166	0	29	1
rls/4CH/regular/single/2	166	0	29	1
rls/APLAX/regular/median/2	166	0	29	1
rls/all-views/scaled/median/2	164	0	29	3
rls/APLAX/normalized/single/2	166	0	29	1
rls/all-views/scaled/single/2	166	0	29	1
rls/APLAX/normalized/average/2	165	0	29	2
rls/4CH/normalized/centroid/2	166	0	29	1
rls/4CH/normalized/median/2	166	0	29	1
rls/APLAX/normalized/centroid/2	166	0	29	1
rls/2CH/regular/weighted/2	166	0	29	1
rls/4CH/z-normalized/median/2	166	0	29	1
rls/4CH/z-normalized/centroid/2	166	0	29	1
rls/2CH/regular/single/2	166	0	29	1
rls/4CH/z-normalized/average/2	166	0	29	1
rls/2CH/regular/average/2	166	0	29	1
rls/4CH/z-normalized/complete/2	166	0	29	1
rls/2CH/regular/centroid/2	166	0	29	1
rls/2CH/normalized/single/2	166	0	29	1
rls/2CH/z-normalized/average/2	166	0	29	1
rls/4CH/z-normalized/single/2	166	0	29	1

rls/2CH/normalized/average/2	166	0	29	1
rls/4CH/normalized/weighted/2	166	0	29	1
rls/2CH/normalized/centroid/2	166	0	29	1
rls/2CH/normalized/median/2	128	0	29	39
rls/2CH/z-normalized/single/2	166	0	29	1
rls/2CH/z-normalized/complete/2	164	0	29	3
rls/all-views/z-normalized/weighted/2	165	0	29	2
rls/APLAX/normalized/median/2	162	0	29	5
gls/APLAX/normalized/average/2	166	0	29	1
gls-rls/all-views/z-normalized/single/2	166	0	29	1
gls-rls/all-views/normalized/single/2	166	0	29	1
gls/APLAX/z-normalized/average/2	166	0	29	1
gls-rls/all-views/normalized/average/2	165	0	29	2
gls-rls/all-views/normalized/ward/2	128	0	29	39
gls-rls/all-views/normalized/centroid/2	166	0	29	1
gls-rls/all-views/normalized/median/2	166	0	29	1
gls-rls/all-views/normalized/weighted/2	166	0	29	1
gls/APLAX/z-normalized/single/2	166	0	29	1
gls-rls/all-views/regular/median/2	166	0	29	1
gls-rls/all-views/z-normalized/average/2	165	0	29	2
gls/APLAX/normalized/weighted/2	166	0	29	1
gls-rls/all-views/z-normalized/centroid/2	166	0	29	1
gls-rls/all-views/z-normalized/weighted/2	165	0	29	2
gls-rls/all-views/scaled/single/2	166	0	29	1
gls/APLAX/normalized/median/2	166	0	29	1
gls/APLAX/normalized/centroid/2	166	0	29	1
gls/APLAX/z-normalized/centroid/2	166	0	29	1
rls/all-views/regular/single/2	166	0	29	1
rls/APLAX/normalized/weighted/2	166	0	29	1
rls/APLAX/scaled/single/2	166	0	29	1
rls/APLAX/z-normalized/single/2	166	0	29	1
rls/all-views/z-normalized/median/2	166	0	29	1
rls/APLAX/z-normalized/average/2	165	0	29	2
rls/all-views/z-normalized/centroid/2	166	0	29	1
rls/APLAX/z-normalized/centroid/2	166	0	29	1
rls/APLAX/z-normalized/median/2	166	0	29	1
rls/APLAX/z-normalized/weighted/2	166	0	29	1
rls/all-views/z-normalized/average/2	165	0	29	2
rls/all-views/regular/weighted/2	166	0	29	1
rls/all-views/z-normalized/single/2	166	0	29	1
rls/all-views/normalized/median/2	166	0	29	1
rls/APLAX/scaled/median/2	166	0	29	1
rls/all-views/normalized/centroid/2	166	0	29	1
gls-rls/all-views/regular/single/2	166	0	29	1
rls/all-views/normalized/average/2	166	0	29	1
rls/all-views/normalized/single/2	166	0	29	1
gls/all-views/regular/single/2	166	0	29	1
gls/4CH/regular/single/2	167	1	28	0
gls/4CH/regular/average/2	167	1	28	0
gls/4CH/scaled/single/2	167	1	28	0
gls/4CH/scaled/average/2	167	1	28	0

gls/2CH/regular/single/2	167	1	28	0
gls/2CH/scaled/single/2	167	1	28	0
rls/all-views/regular/centroid/2	167	0	29	0
rls/all-views/scaled/centroid/2	167	0	29	0
rls/4CH/regular/centroid/2	167	1	28	0
rls/4CH/scaled/centroid/2	167	1	28	0
rls/4CH/scaled/weighted/2	167	1	28	0
rls/2CH/regular/median/2	167	0	29	0
rls/2CH/normalized/weighted/2	167	1	28	0
rls/2CH/z-normalized/median/2	167	2	27	0
rls/APLAX/regular/centroid/2	167	0	29	0
rls/APLAX/scaled/centroid/2	167	0	29	0
gls-rls/all-views/z-normalized/median/2	167	0	29	0
gls-rls/4CH/regular/centroid/2	167	1	28	0
gls-rls/4CH/regular/median/2	167	1	28	0
gls-rls/4CH/scaled/centroid/2	167	1	28	0
gls-rls/2CH/normalized/weighted/2	167	1	28	0
gls-rls/2CH/z-normalized/median/2	167	2	27	0
gls-rls/2CH/scaled/median/2	167	0	29	0
gls-rls/APLAX/z-normalized/median/2	167	0	29	0

Table 10.3: Classification results of applying TSC to identify heart failure among patients. The results are sorted in descending order of DOR, although DOR is not included.

Preprocessing-Method	TP	TN	FP	FN
regular/weighted/2	822	1610	85	996
scaled/weighted/2	822	1610	85	996
regular/ward/2	1202	1491	204	616
scaled/ward/2	1202	1491	204	616
regular/complete/2	1133	1515	180	685
scaled/complete/2	1133	1515	180	685
z-norm/complete/2	471	1604	91	1347
z-norm/weighted/2	583	1553	142	1235
norm/ward/2	903	1049	646	915
z-norm/ward/2	1091	845	850	727
norm/complete/2	1704	58	1637	114
norm/weighted/2	1756	4	1691	62
regular/average/2	1816	0	1695	2
scaled/average/2	1816	0	1695	2
regular/centroid/2	1815	0	1695	3
scaled/centroid/2	1815	0	1695	3
z-norm/average/2	1814	0	1695	4
z-norm/centroid/2	1814	0	1695	4
norm/average/2	1809	0	1695	9
norm/centroid/2	1818	1	1694	0

10.1.2 Peak-value Clustering

Table 10.4: Classification results of applying PVC to identify heart failure among patients. The results are sorted in descending order of DOR, although DOR is not included.

Dataset-Method	TP	TN	FP	FN
gls-EF/ward/2	83	63	37	12
gls-EF/complete/2	77	72	28	18
gls-EF/average/2	81	65	35	14
rls-EF/complete/2	83	55	36	14
gls-rls-EF/ward/2	78	55	36	15
gls-rls-EF/complete/2	70	62	29	23
rls-EF/ward/2	58	74	17	39
rls/average/2	61	72	19	36
gls-rls/ward/2	56	71	20	37
rls/ward/2	57	71	20	40
gls/ward/2	59	74	26	36
gls-rls/complete/2	4	90	1	89
rls/complete/2	58	66	25	39
gls-rls/average/2	92	3	88	1
gls/complete/2	16	83	17	79
rls-EF/single/2	96	0	91	1
rls-EF/average/2	96	0	91	1
gls/average/2	0	99	1	95
gls/single/2	0	99	1	95
gls-rls-EF/single/2	92	0	91	1
gls-rls-EF/average/2	92	0	91	1
rls/single/2	97	1	90	0
gls-EF/single/2	1	100	0	94
gls-rls/single/2	93	1	90	0

Table 10.5: Classification results of applying PVC to identify patient diagnoses among patients. The results are sorted in descending order of DOR, although DOR is not included.

Dataset-Method	TP	TN	FP	FN
gls-EF/ward/2	118	30	2	45
rls-EF/complete/2	117	27	2	42
gls-rls-EF/ward/2	112	27	2	43
gls-EF/average/2	114	30	2	49
gls-EF/complete/2	103	30	2	60
gls-rls-EF/complete/2	97	27	2	58
rls/complete/2	81	27	2	78
rls/average/2	78	27	2	81
gls-rls/ward/2	74	27	2	81
rls/ward/2	75	27	2	84
rls-EF/ward/2	73	27	2	86
gls/ward/2	82	29	3	81
gls-rls/average/2	153	2	27	2
gls/complete/2	137	7	25	26
gls-rls/complete/2	150	0	29	5
gls-EF/single/2	162	0	32	1
rls-EF/single/2	158	0	29	1
rls/single/2	158	0	29	1
rls-EF/average/2	158	0	29	1
gls-rls-EF/single/2	154	0	29	1

gls-rls-EF/average/2	154	0	29	1
gls/single/2	163	1	31	0
gls/average/2	163	1	31	0
gls-rls/single/2	155	1	28	0

10.1.3 Neural Network

Table 10.6: Classification results of NN, when trained to predict heart failure among patients. The results are sorted in descending order of DOR, although DOR is not included.

Dataset-Model	TP	TN	FP	FN
gls/4CH/upsampled	46	61	39	53
rls/APLAX/regular	48	58	42	51
rls/4CH/regular	36	68	32	64
gls/APLAX/downsampled	62	40	60	36
gls/2CH/downsampled	60	39	58	39
gls/4CH/downsampled	48	52	48	51
gls/APLAX/regular	48	50	50	51
gls/2CH/regular	57	39	58	43
gls/4CH/regular	61	34	66	39
all-strain/4CH/regular	52	31	69	48
rls/APLAX/downsampled	33	47	53	65
all-strain/all-views/regular	53	27	70	46
rls/2CH/downsampled	30	45	52	69
all-strain/all-views/downsampled	36	36	61	62
gls/APLAX/upsampled	49	24	76	49
rls/2CH/regular	36	34	63	64
gls/2CH/upsampled	58	16	81	41
all-strain/4CH/upsampled	19	54	46	80
all-strain/2CH/downsampled	64	10	87	35
all-strain/APLAX/regular	41	22	78	58
all-strain/all-views/upsampled	25	33	64	73
all-strain/APLAX/downsampled	34	22	78	64
gls/all-views/regular	25	28	69	74
all-strain/2CH/upsampled	51	9	88	48
rls/all-views/downsampled	51	8	89	47
all-strain/4CH/downsampled	35	15	85	64
rls/4CH/upsampled	22	24	76	77
rls/4CH/downsampled	36	13	87	63
rls/APLAX/upsampled	27	16	84	71
rls/all-views/upsampled	13	29	68	85
gls/all-views/upsampled	13	29	68	85
gls/all-views/downsampled	46	6	91	52
rls/all-views/regular	27	13	84	72
rls/2CH/upsampled	32	9	88	67
all-strain/APLAX/upsampled	41	3	97	57
all-strain/2CH/regular	42	0	97	58

Table 10.7: Classification results of NN, when trained to predict patient diagnoses. The results are sorted in descending order of DOR, although DOR is not included.

Dataset-Preprocessing	TP	TN	FP	FN
all-strain/4CH/upsampled	166	0	32	1
all-strain/2CH/regular	168	0	29	0
gls/2CH/regular	168	0	29	0
rls/2CH/regular	168	0	29	0
all-strain/2CH/downsampled	167	0	29	0
all-strain/2CH/upsampled	167	0	29	0
gls/2CH/downsampled	167	0	29	0
gls/2CH/upsampled	167	0	29	0
rls/2CH/downsampled	167	0	29	0
rls/2CH/upsampled	167	0	29	0
all-strain/all-views/regular	167	0	29	0
gls/all-views/regular	167	0	29	0
rls/all-views/regular	167	0	29	0
all-strain/all-views/downsampled	166	0	29	0
all-strain/all-views/upsampled	166	0	29	0
gls/all-views/downsampled	166	0	29	0
gls/all-views/upsampled	166	0	29	0
rls/all-views/downsampled	166	0	29	0
rls/all-views/upsampled	166	0	29	0
all-strain/4CH/regular	168	0	32	0
gls/4CH/regular	168	0	32	0
rls/4CH/regular	168	0	32	0
all-strain/4CH/downsampled	167	0	32	0
gls/4CH/downsampled	167	0	32	0
gls/4CH/upsampled	167	0	32	0
rls/4CH/downsampled	167	0	32	0
rls/4CH/upsampled	167	0	32	0
all-strain/APLAX/regular	167	0	32	0
gls/APLAX/regular	167	0	32	0
rls/APLAX/regular	167	0	32	0
all-strain/APLAX/downsampled	166	0	32	0
all-strain/APLAX/upsampled	166	0	32	0
gls/APLAX/downsampled	166	0	32	0
gls/APLAX/upsampled	166	0	32	0
rls/APLAX/downsampled	166	0	32	0
rls/APLAX/upsampled	166	0	32	0

Table 10.8: Classification results of NN, when trained to predict segment indication. The results are sorted in descending order of DOR, although DOR is not included.

Preprocessing	TP	TN	FP	FN
regular	1364	1274	607	331
downsampled	1255	1390	473	440
upsampled	934	1365	498	761

10.1.4 Peak-value Supervised Classifiers

Table 10.9: Classification results of PVSC, when trained to predict heart failure among patients. The results are sorted in descending order of DOR, although DOR is not included.

Dataset-Model	TP	TN	FP	FN
gls-EF/Gaussian-Process	74	72	27	21
rls-EF/MLP	74	67	23	23
rls-EF/Linear-SVM	73	67	23	24
gls-EF/Ada-Boost	73	72	27	22
gls-EF/Naive-Bayes	72	73	26	23
gls-EF/Linear-SVM	71	74	25	24
rls-EF/Decision-Tree	76	62	28	21
gls-EF/KNN	70	73	26	25
gls-EF/Random-Forest	74	68	31	21
rls-EF/Extra-Trees	77	60	30	20
gls-rls-EF/Naive-Bayes	71	63	27	22
rls-EF/Naive-Bayes	72	65	25	25
rls/Naive-Bayes	73	64	26	24
gls-rls-EF/Linear-SVM	68	66	24	25
gls-rls-EF/Extra-Trees	72	61	29	21
gls-rls/Decision-Tree	71	62	28	22
gls-rls/Naive-Bayes	70	63	27	23
gls-EF/Discriminant-Analysis	67	74	25	28
gls-EF/Extra-Trees	69	72	27	26
gls-rls-EF/Ada-Boost	69	64	26	24
rls/KNN	79	55	35	18
gls-rls-EF/Random-Forest	70	62	28	23
gls-rls/Extra-Trees	72	59	31	21
gls/Gaussian-Process	70	69	30	25
rls/Ada-Boost	74	60	30	23
gls-rls/Ada-Boost	70	61	29	23
gls/Linear-SVM	70	68	31	25
rls/Linear-SVM	73	60	30	24
gls-EF/Decision-Tree	67	71	28	28
rls-EF/KNN	75	57	33	22
gls/Ada-Boost	70	67	32	25
gls/Naive-Bayes	69	68	31	26
gls-rls/Linear-SVM	67	62	28	26
rls/Extra-Trees	74	57	33	23
gls-rls-EF/KNN	69	59	31	24
rls-EF/Ada-Boost	68	63	27	29
rls-EF/Random-Forest	71	60	30	26
rls/Decision-Tree	74	56	34	23
gls-rls-EF/Decision-Tree	66	61	29	27
gls-rls/KNN	71	55	35	22
gls/Discriminant-Analysis	65	69	30	30
gls-rls/Random-Forest	67	59	31	26
gls-rls/MLP	65	61	29	28
gls/KNN	60	73	26	35
rls/MLP	64	64	26	33
rls/Random-Forest	69	58	32	28
rls-EF/Discriminant-Analysis	68	59	31	29

gls/Extra-Trees	64	67	32	31
rls/Discriminant-Analysis	67	59	31	30
gls-rls-EF/MLP	55	67	23	38
gls/Random-Forest	69	60	39	26
rls/Gaussian-Process	69	52	38	28
rls-EF/Gaussian-Process	69	52	38	28
gls-rls-EF/Discriminant-Analysis	57	61	29	36
gls-rls-EF/Gaussian-Process	64	54	36	29
gls/Decision-Tree	62	63	36	33
gls/RBF-SVM	43	76	23	52
gls-rls/Discriminant-Analysis	54	59	31	39
gls-EF/RBF-SVM	9	95	4	86
gls-EF/MLP	42	74	25	53
gls-rls/Gaussian-Process	59	49	41	34
gls/MLP	40	72	27	55
rls/RBF-SVM	97	0	90	0
gls-rls/RBF-SVM	93	0	90	0
rls-EF/RBF-SVM	97	0	90	0
gls-rls-EF/RBF-SVM	93	0	90	0

Table 10.10: Classification results of PVSC, when trained to predict patient diagnoses. The results are sorted in descending order of DOR, although DOR is not included.

Dataset-Model	TP	TN	FP	FN
gls-rls-EF/Ada-Boost	151	22	6	4
gls-rls/KNN	147	23	5	8
rls-EF/Extra-Trees	153	21	7	6
gls-rls-EF/Extra-Trees	150	20	8	5
gls-rls/Extra-Trees	150	20	8	5
gls-rls-EF/KNN	146	23	5	9
rls/Linear-SVM	155	18	10	4
rls-EF/Random-Forest	155	18	10	4
rls/Extra-Trees	154	19	9	5
gls-rls-EF/Linear-SVM	150	19	9	5
rls-EF/Gaussian-Process	150	22	6	9
rls-EF/Linear-SVM	154	18	10	5
rls-EF/KNN	149	22	6	10
rls-EF/Ada-Boost	153	19	9	6
gls-rls-EF/Gaussian-Process	144	22	6	11
rls/KNN	151	20	8	8
gls-rls/Decision-Tree	147	20	8	8
gls-rls/Linear-SVM	149	18	10	6
gls-rls/Random-Forest	148	18	10	7
rls/Random-Forest	154	15	13	5
rls/Ada-Boost	151	18	10	8
rls/Gaussian-Process	147	20	8	12
gls-rls-EF/Decision-Tree	143	20	8	12
gls-rls/Ada-Boost	149	15	13	6
rls/Naive-Bayes	121	25	3	38
gls-rls/Naive-Bayes	117	25	3	38
rls-EF/Naive-Bayes	120	25	3	39

gls-rls-EF/Naive-Bayes	116	25	3	39
gls-EF/Extra-Trees	154	18	13	9
gls-EF/Naive-Bayes	132	26	5	31
gls/Naive-Bayes	137	25	6	26
rls-EF/Decision-Tree	151	15	13	8
gls-rls-EF/Random-Forest	147	15	13	8
gls-rls/Gaussian-Process	142	18	10	13
gls-rls/MLP	145	16	12	10
rls/Decision-Tree	149	15	13	10
gls-EF/Random-Forest	152	16	15	11
gls-EF/KNN	148	18	13	15
rls-EF/MLP	151	11	17	8
gls/Extra-Trees	152	14	17	11
gls-EF/Gaussian-Process	162	2	29	1
gls-EF/Decision-Tree	147	17	14	16
gls/Random-Forest	153	13	18	10
gls/KNN	152	13	18	11
rls/Discriminant-Analysis	157	3	25	2
rls-EF/Discriminant-Analysis	157	3	25	2
gls-rls-EF/MLP	146	10	18	9
rls/MLP	148	11	17	11
gls/MLP	160	4	27	3
gls-EF/Ada-Boost	147	14	17	16
gls/Decision-Tree	147	14	17	16
gls-EF/Discriminant-Analysis	153	10	21	10
gls/Discriminant-Analysis	153	10	21	10
gls/Ada-Boost	147	13	18	16
gls-EF/MLP	158	5	26	5
gls-EF/Linear-SVM	161	2	29	2
gls/Linear-SVM	161	2	29	2
gls/Gaussian-Process	160	2	29	3
gls/RBF-SVM	163	1	30	0
rls/RBF-SVM	159	0	28	0
gls-rls/RBF-SVM	155	0	28	0
gls-rls/Discriminant-Analysis	155	1	27	0
gls-EF/RBF-SVM	163	0	31	0
rls-EF/RBF-SVM	159	0	28	0
gls-rls-EF/RBF-SVM	155	0	28	0
gls-rls-EF/Discriminant-Analysis	155	1	27	0

Bibliography

[1] Wikipedia contributors. *Cardiology*. English. June 25, 2020. URL: https://en.wikipedia.org/wiki/Cardiology.

[2] Peter J. Brockwell and Richard A. Davis. *Time Series: Theory and Methods Second Edition*. Springer-Verlag New York, Inc., 1991.

[3] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. "Time-series clustering - A decade review". eng. In: *Information Systems* 53 (2015), p. 16. ISSN: 0306-4379.

[4] Hien Nguyen et al. "Maximum Pseudolikelihood Estimation for Model-Based Clustering of Time Series Data". eng. In: *Neural Computation* (2017), p. 990. ISSN: 08997667. URL: http://search.proquest.com/docview/1884823978/.

[5] Maria Ruiz-Abellon, Antonio Gabaldon, and Antonio Guillamon. "Dependency-Aware Clustering of Time Series and Its Application on Energy Markets". eng. In: *Energies* 9.10 (2016), p. 809. ISSN: 19961073. URL: http://search.proquest.com/docview/1831861660/.

[6] Joao A Bastos and Jorge Caiado. "Clustering financial time series with variance ratio statistics". eng. In: *Quantitative Finance* 14.12 (2014), pp. 2121–2133. ISSN: 1469-7688. URL: http://www.tandfonline.com/doi/abs/10.1080/14697688.2012.726736.

[7] Jafar Rahmanishamsi, Ali Dolati, and Masoudreza Aghabozorgi. "A Copula Based ICA Algorithm and Its Application to Time Series Clustering". eng. In: *Journal of Classification* 35.2 (2018), pp. 230–249. ISSN: 0176-4268.

[8] Nikhil Buduma. *Fundamentals of deep learning : designing next-generation machine intelligence algorithms*. eng. First edition. Beijing, China: O'Reilly, 2017. ISBN: 1-4919-2560-4.

[9] Wikipedia contributors. *Long short-term memory*. English. June 22, 2020. URL: https://en.wikipedia.org/wiki/Long_short-term_memory.

[10] Wikipedia contributors. *Rand Index*. English. June 22, 2020. URL: https://en.wikipedia.org/wiki/Rand_index.

[11] Sebastian Raschka. *Python machine learning : machine learning and deep learning with Python, scikit-learn, and TensorFlow*. eng. Birmingham, England ; 2017.

[12] Wei-Meng Lee. *Python machine learning*. eng. Indianapolis, Indiana, 2019.

[13] Wikipedia contributors. *Support-vector machine*. English. June 22, 2020. URL: https://en.wikipedia.org/w/index.php?title=Support-vector_machine&oldid=928737848.

[14] Carl Edward Rasmussen. *Gaussian processes for machine learning*. eng. Cambridge, Mass., 2006.

[15] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[16] Susan Athey, Julie Tibshirani, and Stefan Wager. “Generalized random forests”. eng. In: *The Annals of Statistics* 47.2 (2019), pp. 1148–1178. ISSN: 0090-5364.