Tampere University

Babatunde Adewumi

OUTLIER DETECTION USING UNSUPERVISED MACHINE LEARNING ALGORITHMS

Faculty of Information Technology and Communication Sciences
M. Sc. Thesis
July 2025

ABSTRACT

Babatunde Adewumi: Outlier Detection Using Unsupervised Machine Learning Algorithms
M.Sc. Thesis
Tampere University
Master's Degree Programme in Computing Sciences
July 2025

Recently, outlier detection algorithms have been the focus of research in the machine learning community because of the need to improve their performance in various application areas. Most outlier detection tasks are performed by training an unsupervised model on an unlabeled dataset consisting of both outliers and inliers. Outlier detection algorithms have been utilized to identify, for instance, faults in machines, fraud in credit card transactions, and epileptic seizures in electroencephalogram (EEG) data. This thesis aims to leverage unsupervised outlier detection algorithms to detect outliers in two publicly available datasets and to evaluate and compare their performance.

To identify outliers in the datasets (the heart failure clinical records dataset and the epileptic seizure recognition dataset), five different unsupervised outlier detection algorithms were employed: Local Outlier Factor, One-Class Support Vector Machine, Isolation Forest, Minimum Covariance Determinant, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The performance of the algorithms was evaluated and compared using multiple evaluation metrics, including sensitivity, specificity, balanced accuracy, and area under the receiver operating characteristic curve (ROC-AUC).

The results from the evaluation metrics indicate that DBSCAN outperformed the other algorithms in accurately detecting the outliers in the datasets. It accurately detected approximately 77.1% of the outliers in the heart failure clinical records dataset and approximately 95.4% of the outliers in the epileptic seizure recognition dataset. However, it demonstrated inferior performance when identifying the inliers in the datasets. It correctly identified approximately 62.1% and 96.8% of the inliers in the heart failure clinical records dataset and the epileptic seizure recognition dataset, respectively.

Finally, this thesis provides a comprehensive hyperparameter optimization approach for selecting the optimal hyperparameter values for unsupervised outlier detection algorithms and the most efficient feature scaling methods for datasets. Further work could be done to validate the outliers and inliers detected by unsupervised outlier detection models, thereby improving their efficiency and credibility.

Key words and terms: Outlier detection, Unsupervised machine learning algorithms, Local outlier factor, One-class support vector machine, Isolation forest, Minimum covariance determinant, Density-based spatial clustering of applications with noise.

USE OF AI IN THESIS

I have utilised AI tools in my thesis:

☐ No
☒ Yes

The AI tools utilised in my thesis and their purposes are described below:

- **Grammarly (v1.2.169.1689)**: It was used as a writing support to review and correct the entire thesis's spelling, grammar, punctuation, clarity, engagement, and delivery mistakes.
- **ChatGPT-4o**: It was used as an assistant to suggest synonyms for words, provide technical explanations related to the thesis, and answer general questions such as "What are the core components of an abstract?", "What are the alternative phrases for *including but not limited to*?", and "How would you describe the conclusion chapter of a thesis?".
- **Microsoft CoPilot**: It was used to suggest synonyms for words, provide technical explanations related to the thesis, and serve as a coding assistant for implementing the thesis's coding.

I acknowledge that I am fully responsible for the entire content of my thesis, including the parts generated by AI, and accept accountability for any violations of ethical standards in publications.

PREFACE

The past 6 - 5 years have been the most challenging in my life, and it is one of the main reasons why my master’s studies have taken this long. I am grateful to have completed this programme.

I would like to thank my supervisor, Professor Martti Juhola, for his immense patience, guidance, and support throughout this thesis.

I would also like to extend my gratitude to my family, fiancée (Tokunbo), and friends (Richard, Ayoade, and Ayoola) for their encouragement and support.

Tampere, July 2025

Babatunde Adewumi

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

BST	Binary Search Tree
CVDs	Cardiovascular diseases
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
EEG	Electroencephalogram
FastMCD	Fast Minimum Covariance Determinant
HFrEF	Heart Failure with Reduced Ejection Fraction
iForest	Isolation Forest
k-NN	*k*-Nearest Neighbors
LOF	Local Outlier Factor
MCD	Minimum Covariance Determinant
MLE	Maximum Likelihood Estimate
One-Class SVM	One-Class Support Vector Machine
OPTICS	Ordering Points To Identify the Clustering Structure
PCA	Principal Component Analysis
SMO	Sequential Minimal Optimization
SVM	Support Vector Machine
UCI	University of California Irvine

1 Introduction

For many years, researchers have been studying and investigating various algorithms to improve the detection of outliers in data. In recent times, the study of outlier detection techniques has become a significant area of interest within the field of machine learning, driven by the need to detect outliers effectively and accurately for informed decision-making in various application domains. These techniques have been widely used in detecting, for example, fraud in credit card usage, faults in production machines, and epileptic seizures in electroencephalogram (EEG) data.

There are multiple definitions for an outlier because of the different motivations for detecting outliers. Thus, many researchers have developed these definitions in line with their respective research areas, experiences, and perspectives. According to a definition provided by Hawkins [1980, p. 1], an outlier is "an observation that deviates so much from the other observations as to arouse suspicion that it was generated by a different mechanism". On the other hand, an inlier is a data point that behaves similarly to the majority of the data points. In addition to the various definitions of outliers, as noted by Goldstein and Uchida [2016], outliers have two primary characteristics: they have features which are different from those of the normal points, and they are few in number. The terms *outlier* and *anomaly* have been used interchangeably by authors to refer to the same concept.

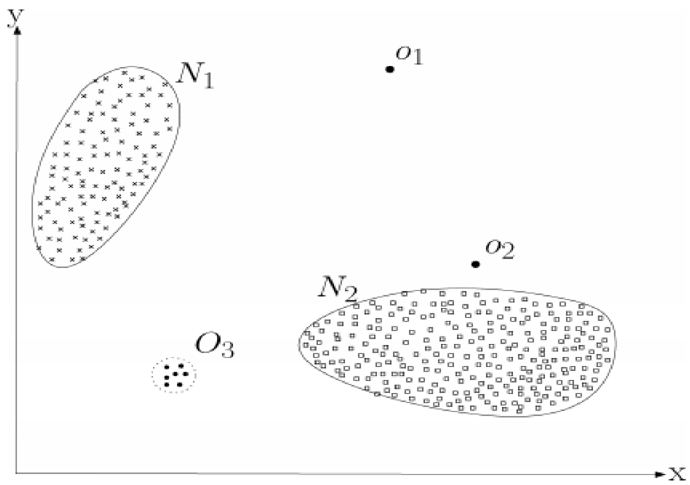

Figure 1.1 *A two-dimensional representation of a dataset showing outliers [Chandola et al., 2009].*

Figure 1.1 shows a two-dimensional illustration of a dataset, where N_1 and N_2 are clusters containing inliers, whereas data points o_1, o_2, and points in O_3 are outliers since they are far apart from the observed normal clusters [Chandola et al., 2009].

Despite the efforts of several researchers to develop more robust outlier detection techniques, detecting outliers in data still presents several challenges. Wang et al. [2019] list some of these challenges as follows:

- The presence of noise in data makes it difficult to correctly identify outliers because the noise usually behaves like the actual outliers.
- The meaning of an outlier is different in various application areas, making it difficult to use an algorithm developed in one area of application in another.
- Absence of clear distinctions between an outlier and an inlier.

The primary aim of this thesis is to employ unsupervised outlier detection algorithms to detect outliers in two publicly available datasets: the heart failure clinical records dataset and the epileptic seizure recognition dataset. The thesis also intends to assess and compare the performance of these algorithms using various evaluation metrics. The remaining part of this thesis proceeds as follows: The second chapter begins by reviewing the different types of outliers, the various approaches to carrying out outlier detection tasks, and lays out the theoretical framework of the outlier detection algorithms. The third chapter mainly focuses on the datasets and their respective explorative analysis, feature scaling methods, and evaluation metrics used in this thesis. The results obtained from the evaluation metrics are presented and described in the fourth chapter, highlighting how the results were achieved, how the hyperparameters of the outlier detection algorithms were optimized, and how the performance comparisons of the outlier detection models were conducted. Lastly, the fifth chapter concludes the thesis by summarizing the main findings and proposing recommendations for future work.

2 Background

This chapter presents an overview of the types of outliers, approaches to carrying out outlier detection tasks, and the theoretical background of the outlier detection algorithms employed in this thesis.

2.1 Types of Outliers

An essential step in identifying outliers in a dataset is to have a good understanding of the types of outliers that may be present in the dataset. This step is crucial for determining the suitability of applying a specific outlier detection technique to a particular dataset. To this end, outliers can be categorized into the following three types:

2.1.1 Point Outliers

In a dataset, a data point is considered a point outlier if it behaves differently from the rest of the data points [Boukerche et al., 2020]. A point outlier is the fundamental type of outlier and is the center of attention in outlier detection research [Chandola et al., 2009]. As an example, the birth rate in Finland is 1.37, but a woman who gave birth to nine children will be considered a point outlier.

2.1.2 Contextual Outliers

According to Chandola et al. [2009], a contextual outlier is a data point that is regarded as an outlier in a particular context, but not an outlier in another context. In other words, a contextual outlier is dependent on its context. In time series data, time is used to determine the context, while longitude and latitude are used to determine the context in spatial data [Chandola et al., 2009]. As a real-life example, a temperature of 25 ºC in summer in Finland can be considered a normal temperature. However, a temperature of 25 °C in January would be regarded as an outlier, because such a temperature value is abnormal in winter in Finland.

2.1.3 Collective Outliers

Chandola et al. [2009] explain that collective outliers are a group of similar data points that collectively show abnormal behavior compared to the rest of the data instances. They further state that each of these data points may not necessarily be an outlier, but their collective existence makes them outliers. In addition, collective outliers only occur in datasets where the data points are correlated with each other [Goldstein and Uchida, 2016]. For example, Figure 2.1 illustrates a region of collective outliers, where the same low value of data instances persists for an unusually long time [Liu et al., 2024]. As a final note, the algorithms used for detecting point and contextual outliers are not suitable for detecting collective outliers. Thus, the outlier detection algorithms used in this thesis are suitable for point outliers, as the chosen datasets only consist of point outliers.

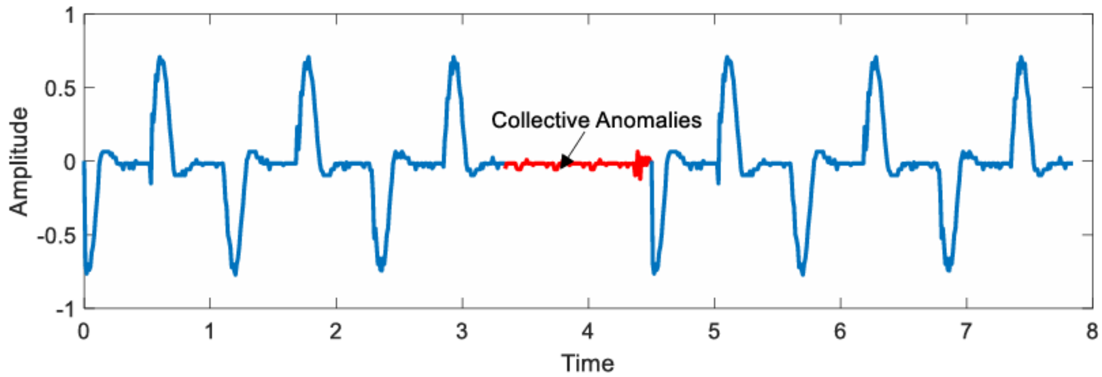

Figure 2.1 *A two-dimensional representation of a dataset showing collective outliers [Liu et al., 2024].*

2.2 Outlier Detection Approaches

The availability of data labels determines how to approach and implement an outlier detection technique. Thus, an outlier detection problem can be approached in one of the following ways:

2.2.1 Supervised Outlier Detection

This approach typically assigns class labels to all data points in both the training and test datasets. Chandola et al. [2009] point out that a training dataset consisting of both the inlier and outlier classes is used to train a predictive model. They further explain that the predictive model is used to predict the classes of the data points present in the unseen test data. This approach for implementing outlier detection techniques is not common like other approaches, due to the difficulty in providing labeled training datasets as well as the problem of class imbalance, which affects the performance of these techniques [Wang et al., 2019].

2.2.2 Semi-supervised Outlier Detection

The outlier detection techniques implemented with this approach also use training and test data samples. However, unlike the techniques in supervised outlier detection, the training dataset only contains labels for the normal data points [Goldstein and Uchida, 2016]. Chandola et al. [2009] note that in rare cases, the training data may consist solely of labeled data points for the outlier class. They furthermore point out that the basic idea of semi-supervised outlier detection involves training a One-Class model with the training dataset and using the model to isolate outliers in the test data sample.

2.2.3 Unsupervised Outlier Detection

This is the most used implementation approach in outlier detection because the algorithms do not require the dataset to be labeled, their performance is not affected by class imbalance, and the dataset only needs to be preprocessed minimally. Here, the dataset consists of both outlier and inlier data instances, and it is assumed that the outliers are only a small proportion of the whole data sample [Chandola et al., 2009]. The basic concept of this approach is to train an unsupervised model with the unlabeled dataset and compute an outlier score for each data point

[Domingues et al., 2018]. These scores are entirely based on the fundamental properties of the data points [Goldstein and Uchida 2016]. In several situations, semi-supervised techniques can be reformulated as unsupervised techniques by utilizing unlabeled datasets that contain both inliers and outliers [Chandola et al., 2009].

2.3 Outlier Detection Algorithm Output

The output of an outlier detection algorithm is used to determine whether a data object is an outlier or not. Depending on the kind of approach used to implement an outlier detection algorithm, the output can be either of two kinds:

2.3.1 Label

This is the so-called binary output, assigned to a data instance to indicate that it is either an inlier or an outlier [Kriegel et al., 2010]. Typically, a label is used to classify data objects in supervised outlier detection [Goldstein and Uchida, 2016].

2.3.2 Score

This is a continuous output, where a score is computed for each data point to indicate the degree of outlierness [Kriegel et al., 2010]. The score is more meaningful and intuitive than a label, and the data instances can be sorted in order of their scores to select the most significant outliers [Goldstein and Uchida 2016; Kriegel et al., 2010]. A threshold can also be defined for the scores to output labels where needed [Kriegel et al., 2010]. Moreover, scores are primarily computed for semi-supervised and unsupervised outlier detection problems [Goldstein and Uchida, 2016].

2.4 Outlier Detection Algorithms

The outlierness of a data point is dependent on the considered reference set within the dataset [Kriegel et al., 2010]. Hence, before examining the various unsupervised outlier detection techniques used in this thesis, it is necessary to distinguish between local and global strategies for finding outliers in datasets. Kriegel et al. [2010] state that the local strategy considers a small proportion of the data points as the reference set and views an outlier as local to its nearby neighborhood. On the other hand, the global strategy considers all other data points and views an outlier as being global rather than local [Kriegel et al., 2010].

Over the years, dozens of unsupervised outlier detection algorithms have been developed, including variants derived from the state-of-the-art algorithms such as Local Outlier Factor (LOF) and *k*-Nearest Neighbors (*k*-NN). As a result, many authors have attempted to categorize these algorithms into different groups based on their perceived fundamental concepts. For example, Chandola et al. [2009] group the algorithms into classification-based, nearest neighbor-based, clustering-based, statistical-based, and information-theoretic-based algorithms, as well as spectral-based algorithms. Wang et al. [2019] categorize the methods into five types: distance-based, density-based, clustering-based, graph-based, and ensemble-based, as well as learning-based methods.

2.4.1 Local Outlier Factor

This is a classical and well-known algorithm that was proposed by Breunig et al. [2000]. The idea of viewing outliers as local to their surrounding neighborhoods was first idealized by these authors. The basic idea of the Local Outlier Factor (LOF) algorithm involves computing a LOF score, which indicates the degree of abnormality of the data instances [Breunig et al., 2000]. In essence, as shown by Breunig et al., the LOF algorithm considers an outlier from a local perspective rather than from a global or binary perspective. Breunig et al. aimed to use the LOF technique to address the problems encountered by distance-based outlier detection techniques when the dataset is more complex and consists of different densities.

As shown by Breunig et al. [2000], the LOF technique utilizes the *k*-nearest neighbors and local reachability density (*lrd*) when calculating the LOF score of an object in a dataset. The authors further indicate that the *lrd* of a data point p is intuitively the inverse of the average reachability distance based on the *k*-nearest neighbors of p. The *lrd* is defined as:

$$lrd_k(p) = \frac{1}{\left(\frac{\sum_{o \in N_k(p)} reach\text{-}dist_k(p,o)}{|N_k(p)|}\right)}, \tag{2.1}$$

where $N_k(p)$ is the *k*-nearest neighbors of p and $reach\text{-}dist_k(p,o)$ is the reachability distance of p with respect to object o, and is given as:

$$reach\text{-}dist_k(p,o) = max\{k\text{-}distance(o), d(p,o)\}, \tag{2.2}$$

where $k\text{-}distance(o)$ is the distance of the *k*-th nearest neighbor of the object o and $d(p,o)$ is the distance between objects p and o.

Finally, as given by Breunig et al. [2000], the LOF score of p, which is the average of the ratio of the local reachability density of p and those of p's *k*-nearest neighbors is computed as shown below:

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|}. \tag{2.3}$$

The value of the LOF score is subsequently used to determine whether an object is an outlier or not. According to Kriegel et al. [2010], if the computed LOF score of the object is approximately equal to 1, then the object is considered a normal data point. On the other hand, if the LOF score is much greater than 1, the object is considered an outlier [Kriegel et al., 2010].

2.4.2 One-Class Support Vector Machine

The One-Class Support Vector Machine (One-Class SVM), a modified version of the classic Support Vector Machine (SVM), was proposed by Schölkopf et al. [2001]. The traditional SVM algorithm, as stated by Shin et al. [2005], seeks to distinguish between two classes of data by finding a hyperplane with a maximum margin between the classes. This, according to the authors, is done by nonlinearly projecting the input data space, X into a high-dimensional feature space and determining the inner product, $\langle \phi(x_i) \cdot \phi(x_j) \rangle$ in the feature space as a function of the original data points. In Figure 2.2, as depicted by Shin et al., the high-dimensional feature space represents a nonlinear transformation of the input data points, allowing a linear function to be used to separate the classes. The resulting linear function in the feature space is given by

$$f(x) = \langle \omega \cdot x \rangle + b, \tag{2.4}$$

where ω is the weight vector and b is the bias [Shin et al., 2005].

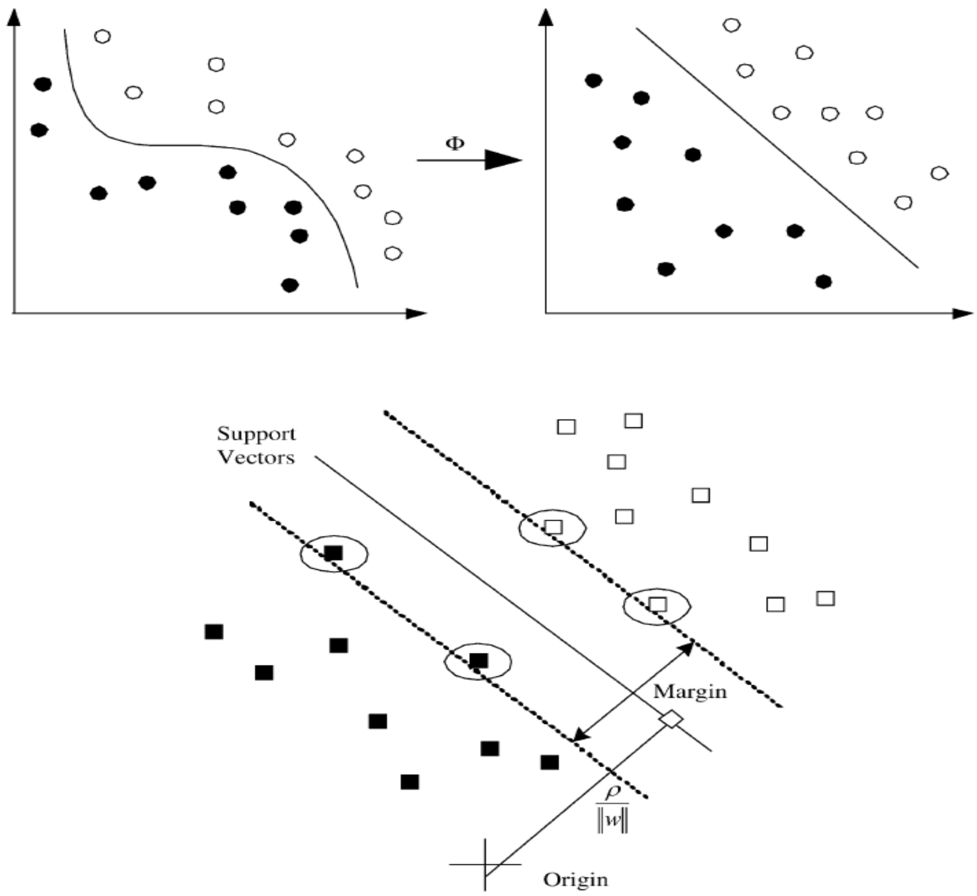

Figure 2.2 *A nonlinear projection of input data points to a high-dimensional space [Shin et al., 2005].*

Unlike the SVM algorithm, the One-Class SVM algorithm tries to separate a class of data points from all other data points, which are then treated as outliers. This is achieved by estimating a nonlinear decision boundary (hyperplane) with a maximum margin that separates the data points from the origin [Amer et al., 2013]. According to Amer et al. [2013], this was the original approach from which the classic SVM algorithm was developed. They state that the approach emerged from Vapnik et al. [1963], but was later abandoned because of the failure to find a nonlinear decision boundary that separates the data points from the origin and to explain outliers. Amer et al. also state that introducing kernels and soft margins resolved these limitations. The One-Class SVM algorithm has been applied successfully to several real-world problems. Shin et al. [2005] applied the algorithm to detect faults in a machine, and Karpov et al. [2023] used it to label epileptic seizures in human EEG. Likewise, Manevitz and Yousef [2001] applied the algorithm to document classification.

As shown by Schölkopf et al. [2001], the One-Class SVM algorithm returns a function f which takes $+1$ in a small region encircling majority of the data points and -1 elsewhere. They achieved this by projecting the input data points into a high-dimensional feature space via a kernel function, particularly the Gaussian kernel, and separating the mapped points from the origin with a maximum margin. Then, the value of $f(x)$ is determined for a new data point x_i by evaluating which side of the decision boundary it falls on in the feature space [Schölkopf et al., 2001]. Figure 2.3 below depicts the outcome of the input data points separation, where the data points in the region with $+1$ are considered normal points, while points, including the origin and the region with -1 are regarded as outliers [Manevitz and Yousef 2001].

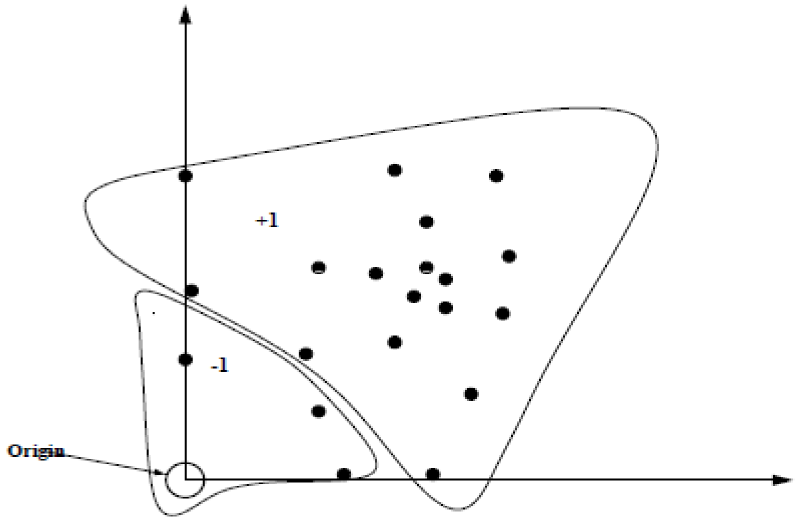

Figure 2.3 *The Outcome of separating input data points using One-Class SVM [Manevitz and Yousef 2001].*

The below quadratic programming problem is solved to separate the input points from the origin:

$$\min_{\omega,\xi_i,\rho} \frac{\| \omega \|^2}{2} + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i - \rho \qquad (2.5)$$
$$\text{subject to: } (\omega \cdot \phi(x_i)) \geq \rho - \xi_i,\ \xi_i \geq 0,$$

where ξ_i are the slack variables used to estimate the separation errors, n is the number of observations in the dataset, ω is the hyperplane normal vector, ρ is the bias term, and ν is the regularization parameter with $\nu \in (0, 1]$ [Schölkopf et al., 2001]. Bounsiar and Madden [2014] state that the transformation function $\phi(.)$ represents the nonlinear mapping of the data points into the feature space and is determined via a kernel function given by

$$k(x_i, x_j) = \langle \phi(x_i) \cdot \phi(x_j) \rangle. \qquad (2.6)$$

According to Schölkopf et al. [2001], if ω and ρ solve the quadratic programming program, then the decision function is given by

$$f(x) = sgn(\omega \cdot \phi(x_i) - \rho). \qquad (2.7)$$

2.4.3 Isolation Forest

Isolation Forest (iForest) is an ensemble-based outlier detection algorithm, proposed by Liu et al. [2012]. Liu et al. point out that, to detect outlier data points, most model-based outlier detection techniques first create a profile of normal data points and then consider those points that are different from the normal profile as outliers. In contrast, they maintain that iForest isolates abnormal data points without creating profiles of normal points or using any distance or density measure. To accomplish this, iForest leverages the basic concept that outliers are more prone to isolation because they are few and different from other data points. Liu et al. implemented the isolation process using a proper binary tree structure known as isolation tree (iTree). Outliers are isolated close to the root of the iTree, while normal data points are isolated near the end of the tree [Liu et al., 2012].

According to Liu et al. [2012], the iForest algorithm constructs an ensemble of randomly generated binary trees, recursively partitioning the data points. The data points are divided into partitions by randomly selecting a feature and randomly picking a split value between the minimum and maximum values of the chosen feature [Liu et al., 2012]. Liu et al. [2012] specified the number of partitions needed for a data point to be isolated as the path length traversed by the point from the root node to the terminated external node on a tree. Outliers are data points that require a smaller number of partitions to be isolated, while normal points require a higher number of partitions to be isolated [Liu et al., 2012]. Furthermore, the path length of a data point, which is used to measure the degree of susceptibility to isolation, is the number of edges the point traverses a tree from the root node to the terminated external node [Liu et al., 2012].

To build the ensembled trees, Liu et al. [2012] specified the number of trees, T, to assemble and the subsampling size ψ of the dataset. Moreover, they demonstrate that iForest has a linear time complexity and can handle very large datasets and those with high dimensionality, thanks to its subsampling capability. In Figure 2.4, the normal data point x_i requires, as expected, more partitions to be isolated [Liu et al., 2012]. On the other hand, the outlier point x_0 requires only a few partitions to be isolated. More specifically, as shown in Figure 2.5, x_0 has a shorter average path length than x_i when the path lengths are averaged over a collection of trees [Liu et al., 2012].

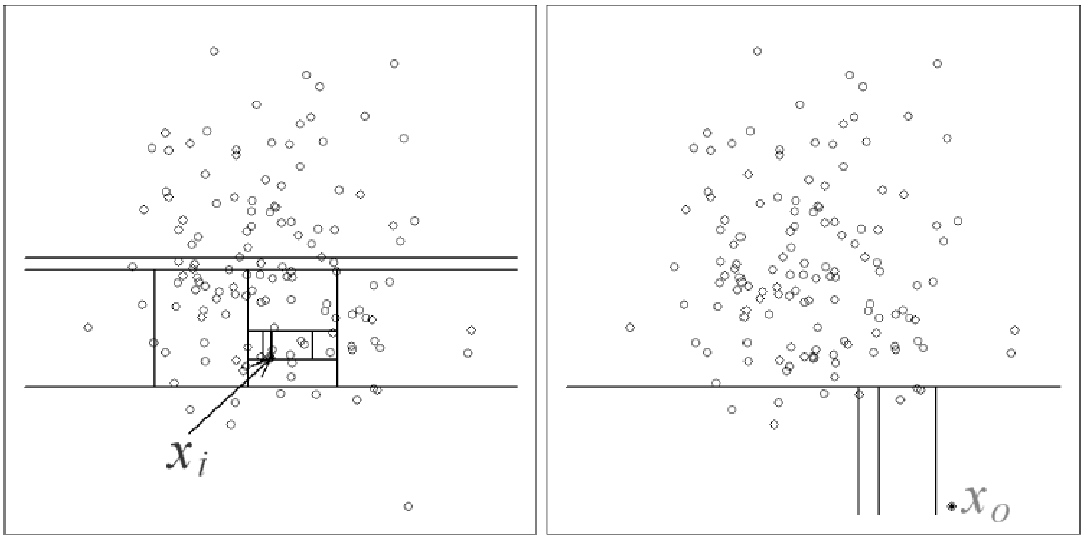

Figure 2.4 *Partitions required to isolate a normal point and an outlier point [Liu et al., 2012].*

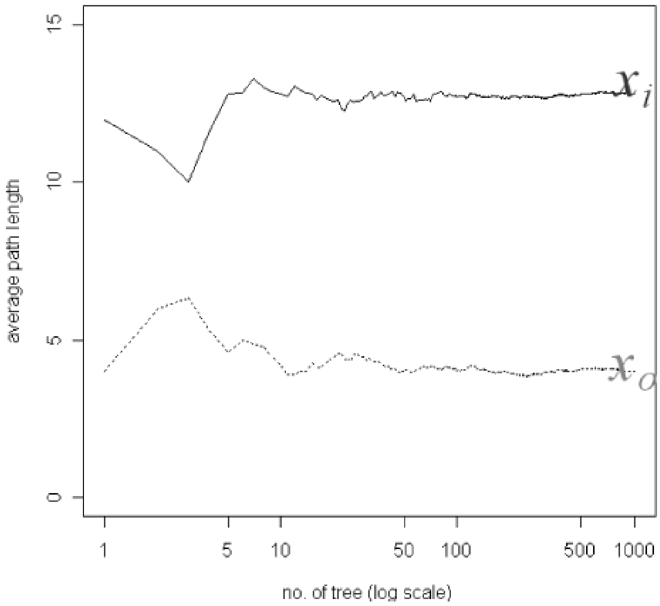

Figure 2.5 *Average path lengths to isolate a normal point and an outlier point [Liu et al., 2012].*

Liu et al. [2012] employed the iForest algorithm to calculate an outlier score for a data point, thereby determining whether the data point is an outlier or not. The outlier score, as defined by Liu et al., is shown below:

$$s(x,\psi) = 2^{-\frac{E(h(x))}{c(\psi)}}, \tag{2.8}$$

where $h(x)$ is the path length of a data point x and $E(h(x))$ is the average of $h(x)$ from a collection of iTrees. In addition, $c(\psi)$ is the average of $h(x)$ given the subsampling size ψ, and it is used to normalize $h(x)$. Specifically, $c(\psi)$ is the average path length of unsuccessful searches in Binary Search Trees (BST), used to estimate the average $h(x)$ for external node termination [Liu et al., 2012]. It is defined as:

$$c(\psi) = \begin{cases} 2H(\psi-1) - 2(\psi-1)/\psi & \text{for } \psi > 2 \\ 1 & \text{for } \psi = 2 \\ 0 & otherwise, \end{cases} \tag{2.9}$$

where $H(i)$ is the harmonic number.

Considering the outlier score s of a data point, if s is very close to 1, then the point is an outlier [Liu et al., 2012]. However, if s is much less than 0.5, it is considered an inlier. Figure 2.6, as depicted by Liu et al. [2012], shows the relationship between $E(h(x))$ and s for $0 < s \leq 1$ and $0 < h(x) \leq \psi - 1$. Three special cases can be deduced from the figure as follows:

- When $E(h(x)) \to 0, s \to 1$.
- When $E(h(x)) \to \psi - 1, s \to 0$.
- When $E(h(x)) \to c(\psi), s \to 0.5$.

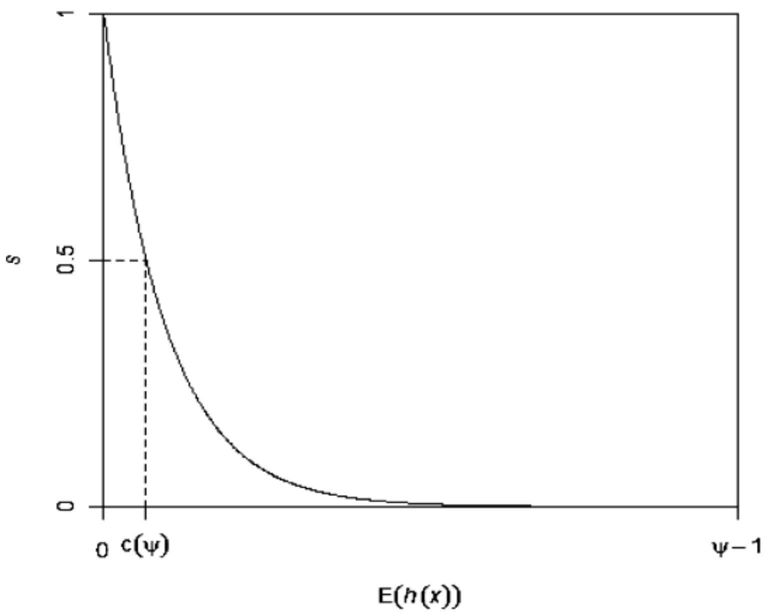

***Figure 2.6** Relationship between E(h(x)) and outlier score [Liu et al., 2012].*

2.4.4 Minimum Covariance Determinant

The Minimum Covariance Determinant (MCD), introduced by Rousseeuw [1984], is an extremely robust estimator of multivariate covariance and location. Unlike the maximum likelihood estimate (MLE) for covariance, the MCD is resistant to outliers, making it effective in detecting outlying data points in datasets [Pedregosa et al., 2011]. The applications of the MCD algorithms in developing robust multivariate methods only began after Rousseeuw [1999] developed the FastMCD algorithm, which is far more computationally efficient than the existing older MCD algorithms. As a result, the MCD has been applied in various domains, including chemistry, finance, image analysis, and medicine [Hubert et al., 2018]. For instance, Fritsch et al. [2012] employed regularized MCD to detect outliers in high-dimensional neuroimaging datasets. Prastawa et al. [2004] utilized MCD to develop a framework for segmenting brain tumors, and Filzmoser et al. [2005] applied Mahalanobis distance-based MCD to identify outlying observations in a geochemical dataset. In addition, the MCD algorithm has been utilized to develop more robust and efficient multivariate statistical methods, including robust Principal Component Analysis (PCA), multiple regression, and factor analysis [Hubert et al., 2018].

In a multivariate dataset, observations are represented by an $n \times p$ matrix $X = (x_1, \dots, x_n)'$ where $x_i = \left(x_{i1}, \dots, x_{ip}\right)$ is the ith observation, n is the number of observations, and p is the number of variables [Hubert et al., 2018]. According to Hubert et al. [2018], to estimate the unknown parameters of the multivariate dataset, it is assumed that the observations are drawn from an elliptically symmetric unimodal distribution with parameters μ and Σ, in which μ is the location vector with p components and Σ the positive-definite $p \times p$ covariance matrix. The density of the elliptically symmetric unimodal distribution as given by the authors is as follows:

$$f(x) = \frac{1}{\sqrt{|\Sigma|}} g\left(d^2(x, \mu, \Sigma)\right), \tag{2.10}$$

where g is a strictly decreasing real function and $d(x, \mu, \Sigma)$ is the Mahalanobis distance of a point x from the center of the distribution, and is defined as:

$$d(x, \mu, \Sigma) = \sqrt{(x - \mu)'\Sigma^{-1}(x - \mu)}. \tag{2.11}$$

The key idea of the robust MCD estimator is to find $h = \frac{n+p+1}{2}$ subset of observations whose covariance has the smallest determinant [Pedregosa et al., 2011]. In addition, it is recommended that n is greater than $5p$ ($n > 5p$) to obtain better estimates [Hubert et al., 2018]. The MCD estimator is a high breakdown point estimator as it can be used to estimate the covariance of strongly contaminated datasets with up to $\frac{n-p-1}{2}$ outliers [Pedregosa et al., 2011].

To demonstrate the robustness of the MCD estimator against outliers, Hubert et al. [2018] utilized a sample from the wine dataset, as described in Hettich and Bay [1999], which consisted of only two constituents (variables). Figure 2.7 shows a scatter plot of the wine data points with two ellipses formed by the MCD-based and the MLE-based Mahalanobis distances of these data points [Hubert et al., 2018]. The outlying data points are located on the lower right-hand side of the plot. In the figure, the classical tolerance ellipse, which is defined by the MLE-based Mahalanobis distance, encircles almost all the data points. More specifically, as Figure 2.8 shows, none of the distances of the data points are unusually large, and so the estimator is only able to detect three outliers in the dataset [Hubert et al., 2018].

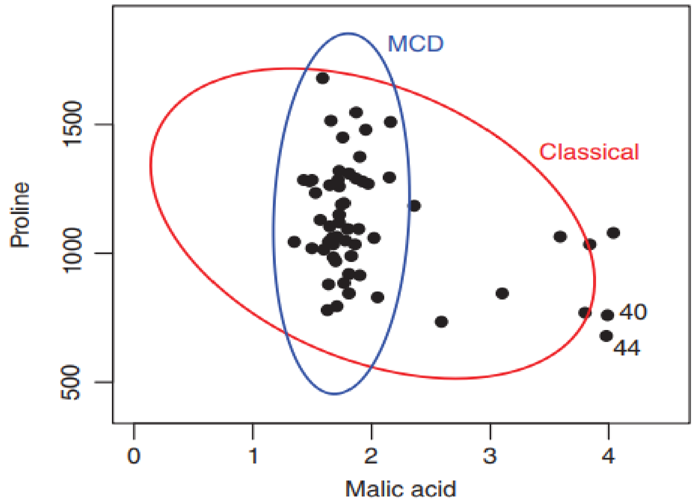

Figure 2.7 *Tolerance ellipses of the classical estimator and that of the robust estimator for the bivariate wine data [Hubert et al., 2018].*

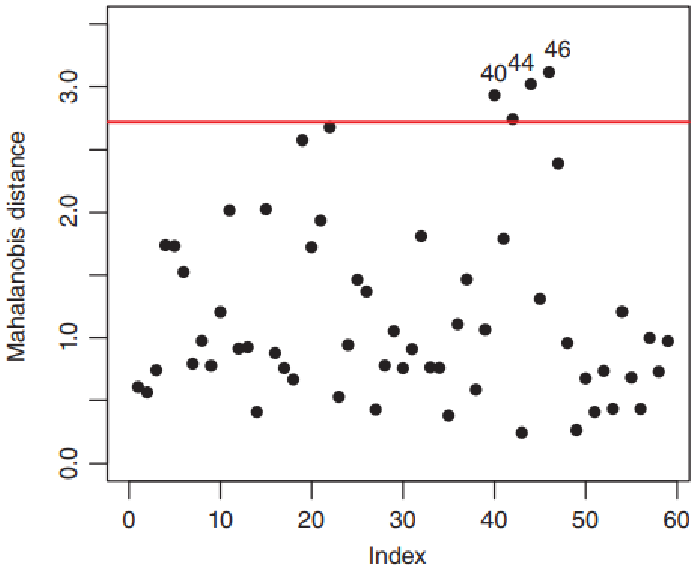

Figure 2.8 *Mahalanobis distances for the bivariate wine data [Hubert et al., 2018].*

On the contrary, as shown in Figure 2.7, the MCD tolerance ellipse, which is defined by the robust MCD Mahalanobis distance, encircles only the normal data points [Hubert et al., 2018]. Furthermore, as illustrated in Figure 2.9, the robust distances of some data points are abnormally large, and therefore, eight data points are identified as outliers [Hubert et al., 2018].

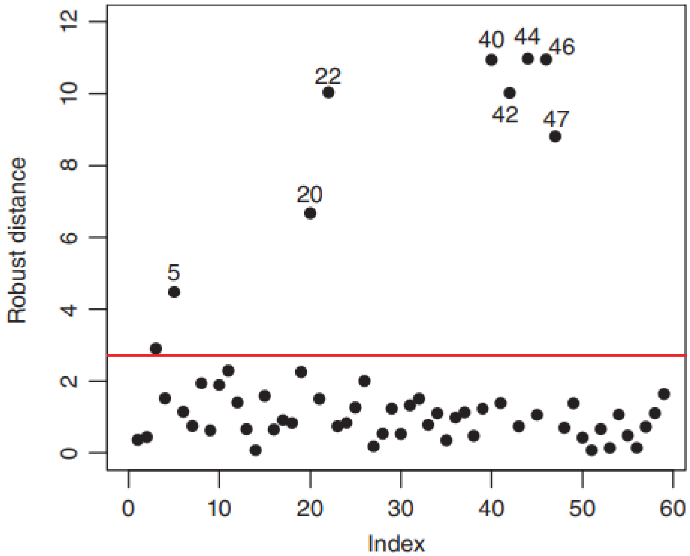

Figure 2.9 *Robust distances for the bivariate wine data [Hubert et al., 2018].*

2.4.5 Density-Based Spatial Clustering of Applications with Noise

Clustering can be viewed as an implicit unsupervised classification process that groups a set of data objects or points into clusters, allowing for the identification of noise or outliers in the data. The objects in the same cluster are identical but remarkably different from those in other clusters [Han et al., 2012]. A Clustering algorithm aims to identify unknown groups or classes in a dataset by evaluating the attribute values of the objects in the data [Han et al., 2012]. According to Han et al. [2012], clustering algorithms can be fundamentally categorized into the following categories:

1. Partitioning algorithms: These algorithms seek to partition a set of n data points into K partitions or clusters where K is the input parameter for the algorithms and $K \leq n$ [Han et al., 2012]. To determine the value of K, some application specific knowledge is needed, which is typically unknown for most applications [Ester et al., 1996]. For a given value of K, a partitioning algorithm begins with an initial dataset partitioning and subsequently employs an iterative control approach to optimize an objective function [Ester et al., 1996]. This partitioning is identical to a Voronoi diagram, with its clusters located in the Voronoi cells, hence, the constructed clusters are convex or spherical, making partitioning algorithms ineffective in detecting outliers [Ester et al., 1996]. Examples of clustering methods in this category are K-Means and K-Medoids algorithms.
2. Hierarchical algorithms: Basically, a hierarchical clustering algorithm builds a hierarchical decomposition (dendrogram) of a set of data points [Ester et al., 1996]. Depending on how the hierarchical decomposition is created, a hierarchical algorithm

is either agglomerative or divisive. The agglomerative technique starts by treating each observation as a separate cluster. Then, the algorithm continuously merges clusters that are closest to each other until all the clusters are merged into one cluster [Han et al., 2012]. The divisive approach, on the other hand, begins with all the data points in one cluster [Han et al., 2012]. In subsequent steps, the cluster is divided into smaller clusters until each data point is assigned to a unique cluster [Han et al., 2012]. Unlike partitioning algorithms, hierarchical algorithms do not require any application specific knowledge, thus, the number of clusters, K is not needed. However, a termination condition must be determined to signify when the merging or splitting should stop [Ester et al., 1996].

3. Density-based algorithms: Density-based clustering algorithms are based on the concept of density [Han et al., 2012]. Here, clusters are simply dense areas of data points, isolated by low-density areas [Han et al., 2012]. Intuitively, the neighborhood of a given radius must have at least a minimum number of data points for any given point in a cluster [Ester et al., 1996]. Unlike partitioning algorithms, density-based algorithms do not need the number of clusters K to be specified beforehand and can discover arbitrary-shaped clusters, making them more effective than hierarchical and partitioning algorithms in detecting outliers or noise [Ester et al., 1996]. Figure 2.10 illustrates two well-formed, arbitrary-shaped clusters discovered by a density-based algorithm [Han et al., 2012]. Typical examples of density-based algorithms include DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and OPTICS (Ordering Points To Identify the Clustering Structure) [Han et al., 2012].

Figure 2.10 *Arbitrary-shaped clusters formed by a density-based algorithm [Han et al., 2012].*

4. Grid-based algorithms: In grid-based algorithms, the data area is partitioned or quantized into a definite number of rectangular cells to form a grid structure [Han et al., 2012]. The grid structure on which all the clustering operations are performed, is implemented with a multi-resolution grid data structure [Han et al., 2012].

DBSCAN is a density-based clustering algorithm that is based on the view that clusters are regions of high density and areas of noise are regions of low density, which can be treated as outliers [Ester et al., 1996]. The number of data points close to (within the neighborhood of a given radius) a data point p determines the density of p [Han et al., 2012]. In DBSCAN, core data points and points within their neighborhoods are grouped to form dense regions, which are then clustered [Han et al., 2012]. To clearly understand the intuition behind DBSCAN, the following definitions and concepts are introduced:

1. ε - neighborhood: This is a user-defined parameter used to specify the radius of a neighborhood of a point p, and is given by Ester et al. [1996] in the equation below. If ε is chosen too small, most points will not be clustered, and as a result, they will be regarded as noise. However, if it is chosen too large, almost all the points in the data will be assigned to a single large cluster without any noise being detected.

$$N_\varepsilon(p) \colon \{(q \mid d(p,q) \leq \varepsilon \}. \tag{2.12}$$

2. MinPts: This is another user-defined parameter used to specify the minimum number of points within the ε - neighborhood of a point p [Ester et al., 1996].
3. Core Point: A point is considered a core point if its ε - neighborhood contains at least a minimum number of points ($MinPts$) [Han et al., 2012].
4. Border point: A point is considered a border point if its ε - neighborhood contains less than $MinPts$, but is within the neighborhood of a core point [Ester et al., 1996].
5. Directly density-reachable: Ester et al. [1996] define a point p to be directly density-reachable from a point q with respect to the parameters ε and $MinPts$ if

$$\begin{gathered} p \in N_\varepsilon(q) \text{ and} \\ |N_\varepsilon(q)| \geq MinPts. \end{gathered} \tag{2.13}$$

Precisely, p is directly density-reachable from q if p is in the ε - neighborhood of q and q is a core point [Han et al., 2012]. Figure 2.11 below illustrates the concepts of a core point, border point, and directly density-reachability [Ester et al., 1996].

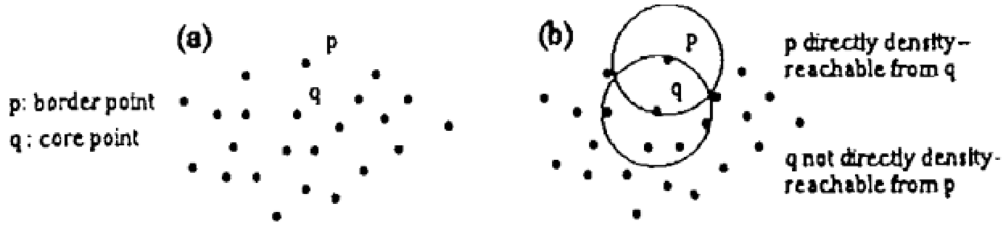

Figure 2.11 *Core points, border points, and directly density-reachability [Ester et al., 1996].*

6. Density-reachable: A point p is density-reachable from a point q with respect to the parameters ε and $MinPts$ if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i [Ester et al., 1996].
7. Density-connected: A point p is density-connected to a point q with respect to the parameters ε and $MinPts$ if there is a point o such that both p and q are density-reachable from o with respect to the parameters ε and $MinPts$ [Ester et al., 1996]. Figure 2.12 illustrates the concepts of density-reachability and density-connectivity [Ester et al., 1996].

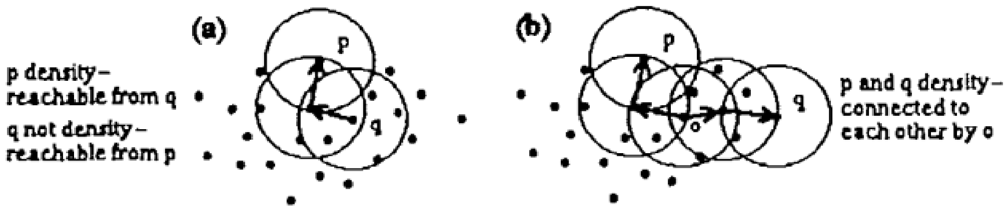

Figure 2.12 *Density-reachability and density-connectivity [Ester et al., 1996].*

8. Noise: Ester et al. [1996] define noise (outliers) as a set of points in the dataset D that do not belong to any cluster and is given as:

$$noise = \{ p \in D \mid \forall i : p \notin C_i \}, \tag{2.14}$$

where C_i are clusters defined with respect to the parameters ε and $MinPts$ and $i = 1, \ldots, K$.

9. Cluster: Ester et al. [1996] define a cluster C with respect to ε and $MinPts$ as a set of density-connected data points and a non-empty subset of the dataset D satisfying the following:
   - Maximality: For all p, q if $p \in C$ and q is density-reachable from p with respect to the parameters ε and $MinPts$, then $q \in C$.
   - Connectivity: For all $p, q \in C$, p is density-connected to q with respect to the parameters ε and $MinPts$.

Conceptually, in the DBSCAN clustering algorithm and given the parameters ε and $MinPts$, clusters are created with the following two steps:

- Initially, a point p is arbitrarily selected from a dataset D satisfying the core point condition as a seed [Ester et al., 1996].
- Then, all points that are density-reachable from p with respect to ε and $MinPts$ are retrieved [Ester et al., 1996].

3 Research Methods

This chapter outlines a detailed approach used in this thesis to detect outliers in two datasets using the various outlier detection algorithms discussed in the previous chapter. It begins with an overview of the datasets, followed by their exploratory analysis. It also presents the preprocessing techniques employed, the evaluation metrics used to assess the performance of the outlier detection methods, and a brief overview of the libraries used in the coding implementation.

3.1 Heart Failure Clinical Records Dataset

Cardiovascular diseases (CVDs) are a category of diseases that affect the heart and blood vessels, encompassing coronary heart disease, cerebrovascular disease, rheumatic heart disease, and other related conditions [World Health Organization, 2024]. According to the World Health Organization [2024], CVDs are the primary cause of death in the world, killing approximately 17.9 million (32% of all global deaths) people every year. The risk of developing CVDs can be reduced by mitigating behavioral and environmental risk factors such as lack of physical activity, use of tobacco, unhealthy diet and obesity, harmful use of alcohol, and air pollution [World Health Organization 2021]. Cardiovascular diseases can lead to heart attack, heart failure, stroke, and sudden cardiac death. A heart attack is when the flow of blood to the heart muscle is abruptly stopped [National Health Service, 2022]. Heart failure is when the heart is not pumping blood properly to the rest of the body [National Health Service, 2022]. A stroke happens when the blood flow to the brain is blocked [National Health Service, 2022].

There is no gainsaying that cardiovascular diseases have become the leading public health problem globally. To this end, early detection and prediction of cardiovascular diseases are crucial to reduce the burden on healthcare systems and the alarming mortality rate due to cardiovascular diseases. The application of machine learning algorithms to medical records can enhance the early detection of CVDs, often before the manifestation of acute symptoms, allowing for early medical intervention and treatment strategies [Naser et al., 2024]. Machine learning can also be used to predict the survival of patients diagnosed with CVDs.

3.1.1 Dataset Description

The heart failure clinical records dataset was obtained from the UCI Machine Learning Repository [2020], and it contains the medical records of 299 patients (194 men and 105 women) aged 40 to 95. The dataset was collected at the Institute of Cardiology and Allied Hospital, Pakistan, in 2015 during the patients' follow-up period (4 to 285 days). As stated by Ahmad et al. [2017], all the patients had left ventricular systolic dysfunction. Left ventricular systolic dysfunction, also known as heart failure with reduced ejection fraction (HFrEF), occurs when the heart is unable to pump blood efficiently [Demolder, 2024]. Of the 299 patients, 96 died of cardiovascular disease. The data consist of 13 clinical features that describe the patients' clinical, lifestyle, and body information [Chicco and Jurman, 2020]. The features consist of seven numerical and six binary features, one of which is the target variable. Also, the dataset has no missing values. Figure 3.1 shows sample rows from the dataset.

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	death_event
0	75.00	0	582	0	20	1	265000.00	1.90	130	1	0	4	1
1	55.00	0	7861	0	38	0	263358.03	1.10	136	1	0	6	1
2	65.00	0	146	0	20	0	162000.00	1.30	129	1	1	7	1
3	50.00	1	111	0	20	0	210000.00	1.90	137	1	0	7	1
4	65.00	1	160	1	20	0	327000.00	2.70	116	0	0	8	1

Figure 3.1 *Sample data from the heart failure clinical records dataset.*

The features of the dataset are described below:

- age: Age of the patient (Integer, years) [Chicco and Jurman, 2020].
- anemia: Decrease of red blood cells or hemoglobin (Binary, false: 0 and true: 1) [Chicco and Jurman, 2020]. Patients who had less than 36% hematocrit levels were considered anemic.
- creatinine_phosphokinase (CPK): Level of the CPK enzyme in the blood (Integer, mcg/L) [Chicco and Jurman, 2020]. The normal level of CPK is between 10 and 120 mcg/L, with higher values suggesting heart failure, stress, or injury [Chen 2023].
- diabetes: If the patient has diabetes (Binary, false: 0 and true: 1) [Chicco and Jurman, 2020].
- ejection_fraction: Percentage of blood leaving the heart at each contraction (Integer, %) [Chicco and Jurman, 2020]. Ejection fraction measures in percentage the amount of blood pumped out of the heart's left ventricle whenever it contracts [American Heart Association 2023]. A healthy heart's ejection fraction is typically between 50 and 70%, but an abnormally low value may indicate heart failure [Cleveland Clinic 2022].
- high_blood_pressure: If the patient has hypertension (Binary, false: 0 and true: 1) [Chicco and Jurman, 2020].
- platelets: Platelets in the blood (Continuous, kiloplatelets/mL) [Chicco and Jurman, 2020]. The normal platelet count in adults ranges from 150 to 450 kiloplatelets/mL [National Heart, Lung, and Blood Institute, 2022]. A platelet count lower than 150 kiloplatelets/mL can lead to excessive bleeding [National Heart, Lung, and Blood Institute, 2022].
- serum_creatinine: Level of serum creatinine in the blood (Continuous, mg/dL) [Chicco and Jurman, 2020]. This variable measures the level of creatinine in the blood. Normal levels of serum creatinine range from 0.9 to 1.3 mg/dL [Stephens 2023]. Abnormally high levels of serum creatinine could indicate renal dysfunction [Stephens 2023].
- serum_sodium: Level of serum sodium in the blood (Integer, mEq/L) [Chicco and Jurman, 2020]. This test measures the amount of sodium in the blood. The normal levels of sodium in the blood are between 135 to 145 mEq/L, but abnormally low values could be a sign of kidney disease or heart failure [Case-Lo 2018].

- sex: Woman or man (Binary, woman: 0 and man: 1) [Chicco and Jurman, 2020].
- smoking: If the patient smokes or not (Binary, false: 0 and true: 1) [Chicco and Jurman, 2020].
- time: Follow-up period of 4 to 285 days (Integer, days) [Chicco and Jurman, 2020].
- death_event (target): If the patient died during the follow-up period (Binary, survived: 0 and died: 1) [Chicco and Jurman, 2020].

3.1.2 Descriptive Statistics

To gain insight into some hidden characteristics of the dataset, descriptive statistics were generated for all the dataset variables. Figure 3.2 presents the summary statistics of the numerical variables, and Table 3.1 compares the mean values of these variables for the patients who survived and those who died.

	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium	time
count	299.00	299.00	299.00	299.00	299.00	299.00	299.00
mean	60.83	581.84	38.08	263358.03	1.39	136.63	130.26
std	11.89	970.29	11.83	97804.24	1.03	4.41	77.61
min	40.00	23.00	14.00	25100.00	0.50	113.00	4.00
25%	51.00	116.50	30.00	212500.00	0.90	134.00	73.00
50%	60.00	250.00	38.00	262000.00	1.10	137.00	115.00
75%	70.00	582.00	45.00	303500.00	1.40	140.00	203.00
max	95.00	7861.00	80.00	850000.00	9.40	148.00	285.00

Figure 3.2 *Summary statistics of numerical variables.*

Numerical variable	Survived mean	Dead mean
age	58.76	65.22
creatinine_phosphokinase	540.05	670.20
ejection_fraction	40.27	33.47
platelets	266657.49	256381.04
serum_creatinine	1.18	1.84
serum_sodium	137.22	135.38
time	158.34	70.89

Table 3.1 *Mean values of numerical variables for the survived and dead patients.*

Table 3.2 below presents the value counts of the binary variables in the dataset. It also details the percentage of each binary value in each binary variable. For example, considering the variable, *anemia*, the number and percentage of patients who were not anemic (170 patients, 56.86% of all patients) and survived were 120 and 70.6%, respectively. However, the number and percentage of patients who were not anemic but died were 50 and 29.4%, respectively. Likewise, the number and percentage of patients who were anemic (129, 43.14% of all the patients) but survived were 83 and 64.34%, respectively. On the other hand, the number and percentage of patients who were anemic and died were 46 and 35.66%, respectively.

Binary variable	Entire dataset		Survived		Dead	
	#	%	#	%	#	%
anemia (false: 0)	170	56.86	120	70.60	50	29.40
anaemia (true: 1)	129	43.14	83	64.34	46	35.66
diabetes (false: 0)	174	58.20	118	67.80	56	32.20
diabetes (true: 1)	125	41.80	85	68	40	32
high_blood_pressure (false: 0)	194	64.90	137	70.62	57	29.38
high_blood_pressure (true: 1)	105	35.10	66	62.86	39	37.14
sex (false: 0)	105	35.10	71	67.62	34	32.38
sex (true: 1)	194	64.90	132	68.04	62	31.96
smoking (false: 0)	203	67.90	137	67.50	66	32.50
smoking (true: 1)	96	32.10	66	68.75	30	31.25

Table 3.2 *Binary value counts and percentages of binary variables.*

Furthermore, to detect outliers in the dataset, a data point with death_event (target) variable value of 0 was considered an inlier. Conversely, a data point having death_event value of 1 was treated as an outlier. Therefore, of the 299 observations, 203 were inliers, and 96 were outliers, accounting for less than a third (32%) of the dataset. Figure 3.3 below illustrates the graphical representation of the number of patients who died and survived in the heart failure clinical records dataset.

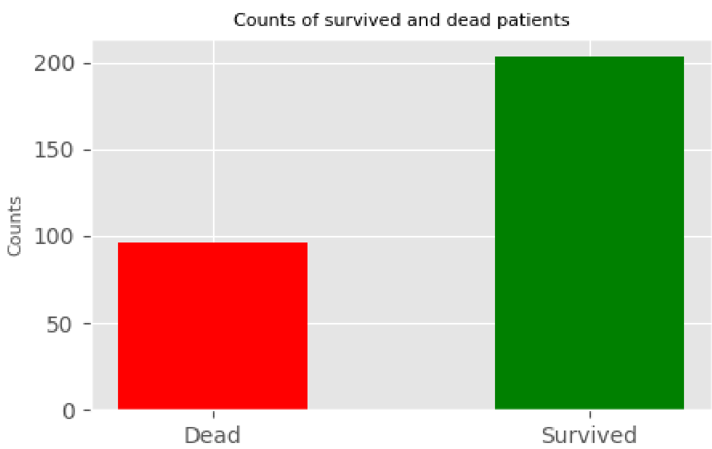

Figure 3.3 *Number of dead and survived patients.*

Lastly, the correlation matrix in Figure 3.4 was constructed to understand some underlying relationships between the variables in the dataset. Some of the highlights from the figure are presented below.

1. It is apparent that time has a negative correlation (-0.53) with death_event, suggesting that patients who had a shorter follow-up period are more likely to die or to have died.

2. A weak positive correlation (0.25) was found between age and death_event, implying that older patients were marginally likely to die. However, the correlation was weak, and the other variables also affected the likelihood of death.
3. There was a negative correlation (-0.27) between ejection_fraction and death_event, suggesting that patients with lower ejection_fraction were slightly likely to die. Nonetheless, the correlation was weak, and the other variables also influenced the tendency to die.
4. There was also a weak positive correlation (0.29) between serum_creatinine and death_event. This means that patients with higher serum_creatinine were somewhat likely to die. Nevertheless, the correlation was weak, and other variables also influenced the likelihood of death.
5. It can also be seen from the figure that there was a weak negative correlation (-0.2) between serum_sodium and death_event, meaning patients with lower serum_sodium were slightly more likely to die. However, the correlation was weak, and the other variables also influenced the possibility of death.

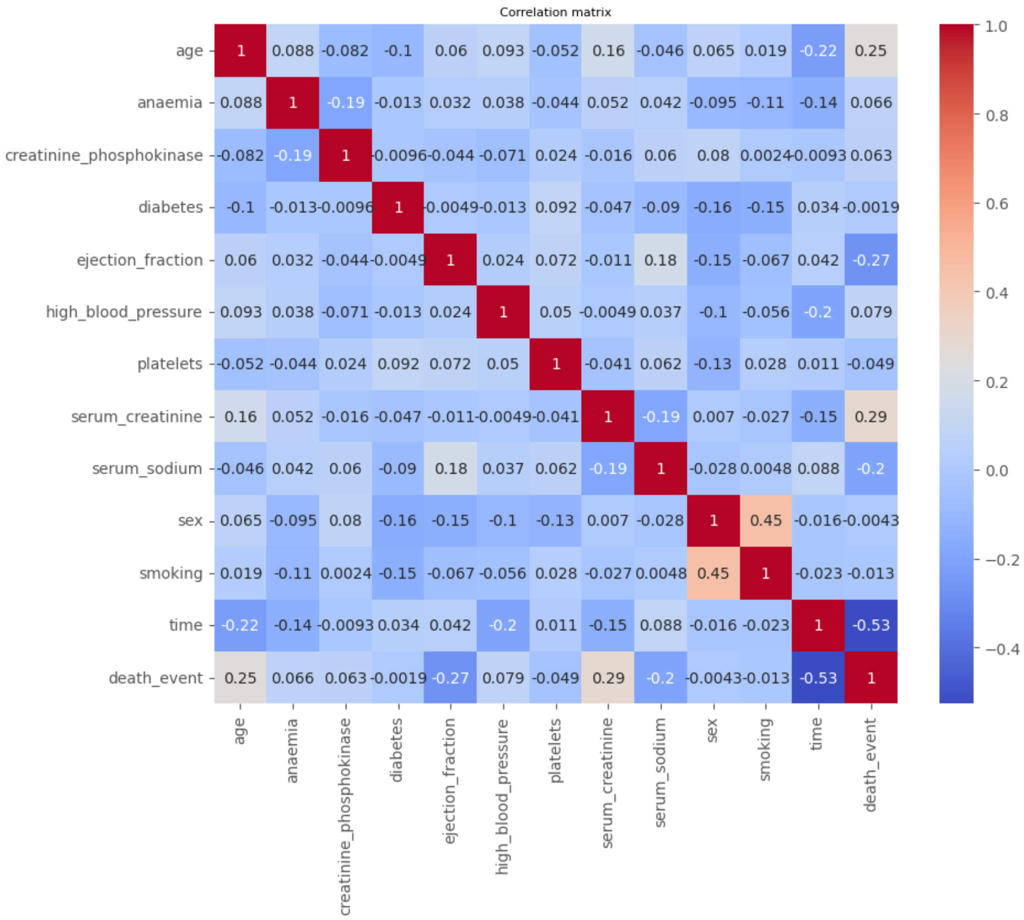

Figure 3.4 *Correlation matrix of the heart failure clinical records dataset.*

3.2 Epileptic Seizure Recognition Dataset

Epilepsy is one of the most protracted neurological diseases caused by recurring seizures and characterized by sudden and uncontrollable discharge of abnormal electrical signals from the brain [Kode et al., 2024]. Although in about 70% of the cases, the cause of epileptic seizures is not known, they can be caused by factors such as genetics, head injuries, developmental disorders, as well as immune disorders [Cleveland Clinic 2022]. Epilepsy can affect anyone of any age, and it has a prevalence of around 0.6 - 0.8% of the world's population [Mormann et al., 2007]. Epileptic seizures can leave both physical and mental symptoms on the patients [Kode et al., 2024]. These symptoms, depending on the type of seizure, include loss of consciousness, uncontrollable jerking and shaking, serious injuries, and confusion [Cleveland Clinic 2022]. Sometimes, these epileptic seizures can also leave long-term psychological effects on the patients and can cause sudden and unexpected death.

Diagnosing epilepsy is very challenging because electroencephalogram (EEG) signals are often degraded by artefactual components that interfere with the actual brain signals [Sai et al., 2021]. The application of machine learning algorithms on EEG signals can be leveraged to remove these artifacts and accurately detect and diagnose epileptic seizures. This will ensure early diagnosis of the disease, improve the quality of life for epilepsy patients, reduce life-threatening risks, and facilitate the formulation of individualized treatment plans.

3.2.1 Dataset Description

Electroencephalogram is a non-invasive and the most common method for detecting and diagnosing epilepsy. It is used to record the electrical activity of the brain by attaching electrodes to the scalp. Typically, EEG signals are preprocessed to remove artefactual components and other irrelevant information before applying machine learning algorithms to detect seizures in the signals. The EEG dataset used in this thesis is the popular Epileptic Seizure Recognition dataset made available on the UCI Machine Learning Repository website by Wu and Fokoue [2017]. It is a preprocessed and restructured version of the Bonn EEG data [Andrzejak et al., 2001]. The dataset consists of five distinct folders, each containing 100 files [Wu and Fokoue 2017]. Each file corresponds to the recording of an individual's brain activity for 23.6 seconds [Wu and Fokoue 2017]. The resulting time series was then sampled into 4097 data points, with each data point representing the value of the EEG recording at a particular time [Wu and Fokoue 2017]. Hence, the dataset consists of EEG recordings from 500 individuals, each with 4097 data points spanning 23.6 seconds [Wu and Fokoue 2017].

Furthermore, Wu and Fokoue [2017] divided and shuffled each of the 4097 data points into 23 chunks, with each chunk having 178 data points. Each of these 178 data points (X1, X2, ..., X178) corresponds to a single row in the dataset and represents the value of the EEG recording for a 1 second interval [Wu and Fokoue 2017]. Thus, there are $23 \times 500 = 11500$ rows and 180 columns in the dataset, with the first column (Unnamed) being the patient ID (not used) and the last column (y) representing the class label to which each row belongs. In addition, the dataset contains no missing data. The variable y in the dataset has the following five possible classes:

- 5 – Eyes opened, which means that when they were recording the EEG signal, the patients had their eyes opened [Wu and Fokoue 2017].
- 4 – Eyes closed, which means that when they were recording the EEG signal, the patients had their eyes closed [Wu and Fokoue 2017].
- 3 – Yes, they identified where the region of the tumor was in the brain and recorded the EEG activity from the healthy brain area [Wu and Fokoue 2017].
- 2 – They recorded the EEG signal from the area where the tumor was located [Wu and Fokoue 2017].
- 1 – EEG recording of seizure activity [Wu and Fokoue 2017].

An epileptic seizure was not recorded in classes 2, 3, 4, and 5 [Wu and Fokoue 2017]. Only class 1 had an epileptic seizure recording. In the next subsection, classes 2, 3, 4, and 5 will be replaced with 0. Figure 3.5 displays some data rows from the Epileptic Seizure Recognition dataset.

	Unnamed	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X170	X171	X172	X173	X174	X175	X176	X177	X178	y
0	X21.V1.791	135	190	229	223	192	125	55	-9	-33	...	-17	-15	-31	-77	-103	-127	-116	-83	-51	4
1	X15.V1.924	386	382	356	331	320	315	307	272	244	...	164	150	146	152	157	156	154	143	129	1
2	X8.V1.1	-32	-39	-47	-37	-32	-36	-57	-73	-85	...	57	64	48	19	-12	-30	-35	-35	-36	5
3	X16.V1.60	-105	-101	-96	-92	-89	-95	-102	-100	-87	...	-82	-81	-80	-77	-85	-77	-72	-69	-65	5
4	X20.V1.54	-9	-65	-98	-102	-78	-48	-16	0	-21	...	4	2	-12	-32	-41	-65	-83	-89	-73	5

5 rows × 180 columns

Figure 3.5 *Sample data from the epileptic seizure recognition dataset.*

3.2.2 Dataset Visualization

To understand the dataset, it is essential to highlight the EEG signal patterns present in the dataset, particularly how the EEG signals associated with seizure activity differ from those of non-seizure activity. Moreover, the number of data records corresponding to seizure activity is of utmost importance in detecting outliers in the dataset. Figure 3.6 shows the plots of the various EEG signal patterns recorded for all the classes. For simplicity, a sample EEG signal was selected from each class. As can be observed from the figure, the EEG recording from the healthy brain region (y = 3) reveals regular signal patterns with steady amplitude, indicating normal electrical activity. The EEG signal recorded from the tumor region (y = 2) shows variations when compared to the signal from the healthy brain area. The EEG signals recorded when patients' eyes were closed (y = 4) and opened (y = 5) exhibit patterns that differ from those exhibited by the EEG signals recorded from the healthy brain area. Additionally, these signals exhibit more frequent fluctuations in amplitude. What stands out in the plots is the very distinctive patterns shown by the EEG recording of seizure activity (y = 1). This signal indicates abnormal electrical activity and is characterized by high amplitude.

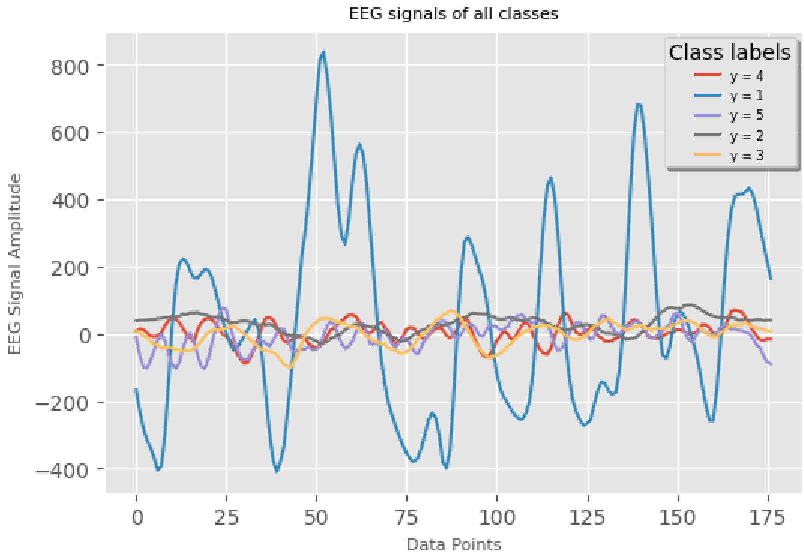

Figure 3.6 *Sample EEG recordings of all the classes.*

For a better comparison of the seizure and non-seizure recordings, and to identify seizure activity in the dataset, all non-seizure class labels were replaced with 0, resulting in only two classes: 0 (non-seizure) and 1 (seizure). Figure 3.7 shows the EEG signal patterns from these classes. Again, a sample was selected from each of the classes. As can be seen from the figure, the EEG signal of the non-seizure class (y = 0) reflects normal electrical brain activity. In contrast, the signal of the seizure class (y = 1) exhibits abnormal electrical brain activity and is characterized by high amplitude.

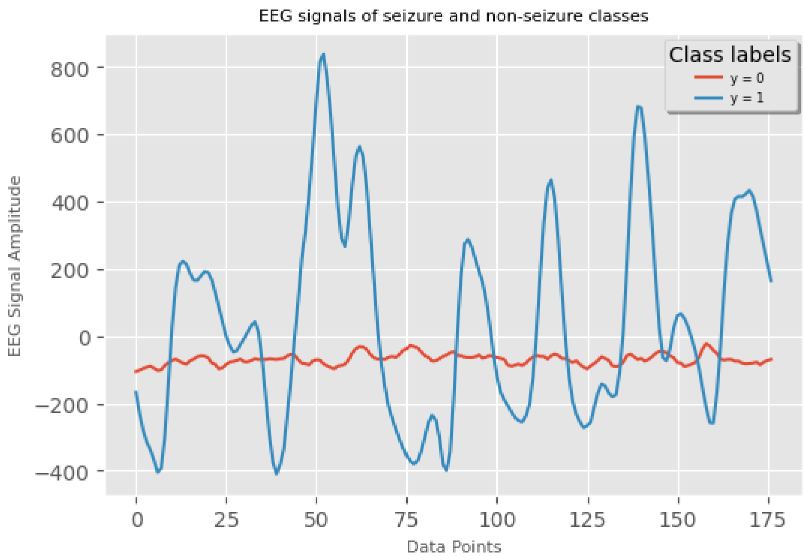

Figure 3.7 *Sample EEG signals of seizure and non-seizure classes.*

Furthermore, Figure 3.8 shows the number of data points from the seizure and non-seizure classes in the dataset. There were 2300 data points belonging to class 1 and 9200 data points belonging to class 0. For the sake of identifying outliers in the dataset, samples with class label 0 represent inliers while those with class label 1 represent outliers. Thus, out of the 11500 data points in the dataset, 9200 observations were inliers, while 2300 were outliers (20% of the dataset).

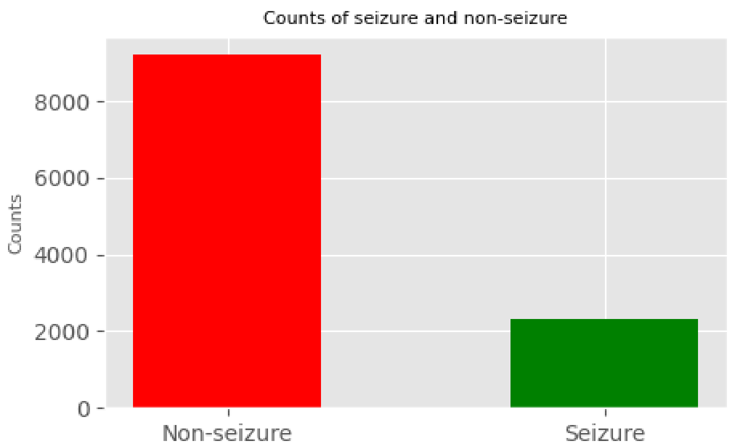

Figure 3.8 *Number of seizure and non-seizure data points.*

3.3 Feature Scaling Methods

Feature scaling is an important preprocessing step for machine learning models. When preparing data for machine learning algorithms, the dataset typically contains features with different reference scales, making them non-comparable to each other. For instance, in the heart failure clinical records dataset, age has values ranging from 40 to 95, while platelet count varies between 25100 and 850000. Platelet count will mostly influence a distance measure computed on these different scales. To avoid this scenario, feature scaling can be used to transform the data so that all features are comparable on a similar scale and contribute equally to the machine learning algorithm's learning process.

There are several reasons why features are rescaled before fitting a machine learning model with data: to improve model predictive performance, to expedite the convergence of gradient-based models, and to enable comparison of the model fit with rescaled data to the model fit with unscaled data [Pedregosa et al., 2011]. The following feature scaling techniques were employed to rescale the datasets used in this thesis before fitting the various outlier detection models with the datasets.

3.3.1 Standardization

Standardization, also known as z-score normalization, transforms each feature individually such that it has a mean of 0 and a standard deviation of 1 [Pedregosa et al., 2011]. This method is common because many machine learning algorithms assume each feature behaves like standard normally distributed data [Pedregosa et al., 2011]. Furthermore, standardization is sensitive to outliers, as both the mean and the standard deviation are affected by marginal and extreme outlier values [Pedregosa et al., 2011]. Feature scaling by standardization is defined by:

$$x' = \frac{x - \mu}{\sigma}, \tag{3.1}$$

where μ is the mean and σ is the standard deviation [Han et al., 2012].

3.3.2 MinMax Scaler

This feature scaling technique rescales the values of each feature individually to vary between a given minimum and maximum value, usually between zero and one [Pedregosa et al., 2011]. It is sensitive to outliers, however, it scales down extreme values so that the smallest data point maps to the minimum value and the largest point maps to the maximum value [Pedregosa et al., 2011]. Generally, the MinMax scaler is given as:

$$x' = a + \frac{(x - x_{min})(b - a)}{x_{max} - x_{min}}, \tag{3.2}$$

where x_{min} and x_{max} are respectively the minimum and maximum values in each feature, while a and b are values corresponding to the feature range [Han et al., 2012].

3.3.3 Robust Scaler

Robust scaler transforms the values of each feature individually by subtracting the median and then dividing by the interquartile range (IQR) [Pedregosa et al., 2011]. The IQR is the range between the 25th (1st quartile) and 75th (3rd quartile) percentiles. This scaling technique is robust to outliers, as the median and the IQR are less influenced by extreme values than the mean and the standard deviation [Pedregosa et al., 2011]. Unlike standardization, the robust scaler does not compress marginal outlier values [Pedregosa et al., 2011]. The mathematical formula for a robust scaler is given as:

$$x' = \frac{x - Q_2(x)}{Q_3(x) - Q_1(x)}, \tag{3.3}$$

where $Q_1(x)$ is the 1st quartile, $Q_2(x)$ is the median (2nd quartile), and $Q_3(x)$ is the 3rd quartile [Khoirunnisa and Ramadhan 2023].

3.3.4 Normalizer

This method scales each sample (each row of the data) rather than each feature so that its norm is 1 [Pedregosa et al., 2011]. The transformation is done independently of the distribution of the samples and is defined by:

$$x' = \frac{x}{\|x\|}, \tag{3.4}$$

where $\|x\|$ is the Euclidean norm of the sample vector [Pedregosa et al., 2011].

3.3.5 Spline Transformation

This transformation involves adding nonlinear terms into the data by producing B-spline basis functions for each feature separately. A spline is a piecewise polynomial function parametrized by its polynomial degree and positions of the knots [Pedregosa et al., 2011]. Spline transformation is based on a B-spline basis, and it is given by:

$$C(x) = \sum_{i=0}^{n} P_i N_i(x), \tag{3.5}$$

where P_i are the control points of the B-spline, $N_i(x)$ are the B-spline basis functions and n is the number of knots [Askarin et al., 2025].

3.4 Principal Component Analysis

Principal component analysis (PCA) is a linear dimensionality reduction method used in data visualization, data preprocessing, exploratory data analysis, and image compression. It facilitates data visualization, reduces the training time of machine learning algorithms, and enhances model performance by mitigating the effects of the curse of dimensionality and overfitting. Essentially, PCA finds a low-dimensional representation of high-dimensional data that captures most of the original information. The dimensions (principal components) computed by PCA represent the linear transformations of the original features that capture the highest variance [IBM 2023]. The first principal component (PC1) is the direction of the maximum variance in the feature space [IBM 2023]. The larger the variance explained by the first principal component, the more information is preserved from the original data, and no other principal component can capture a higher variance than PC1 [IBM 2023]. The second principal component (PC2) explains the second highest variability, and it is independent of PC1 [IBM 2023].

For an $n \times p$ data matrix X, with each feature centered to have a mean of 0, using PCA, the reduction of X into its k principal components are given by:

$$Z = XV, \tag{3.6}$$

where V is a $p \times k$ matrix with $k < p$ and columns that are the top k eigenvectors of the covariance matrix, Σ of X, and Z is the resulting data matrix from the transformation of X into its new k-dimensional feature space.

3.5 Evaluation Metrics

An outlier detection problem can be regarded as a binary classification problem since it involves classifying data points as either inliers or outliers. Hence, the common metrics used to measure the performance of binary classification algorithms can be used to measure and compare the performance of outlier detection algorithms. Choosing appropriate metrics to evaluate model performance is challenging, and it is particularly complicated when there is an imbalanced class distribution, as seen in outlier detection problems. To address this, a combination of different evaluation metrics was employed in this thesis.

3.5.1 Confusion Matrix

A confusion matrix is an $n \times n$ matrix (table) used to visualize the performance of a classification model, where n is the number of target classes. Each row of a confusion matrix represents the true class, while each column corresponds to the predicted class. The diagonal elements of the matrix are the number of observations that are correctly predicted, while the off-diagonal elements are the number of observations that are incorrectly classified. A confusion matrix provides a more detailed analysis of a classifier's performance by comparing the true classes with the predicted classes. In this thesis, a two-class (2×2) confusion matrix was used to analyze and compare the performance of the outlier detection models, where class 0 was regarded as the positive class (majority class) and class 1 treated as the negative class (minority class).

Figure 3.9 below illustrates a two-class confusion matrix showing all four possible outcomes [Branco et al., 2016]. True positives (TP) are the observations that were correctly identified as positive, and false negatives (FN) are the data points that were incorrectly predicted as negative [Branco et al., 2016]. Likewise, the true negatives (TN) are the observations that were correctly identified as negative, and false positives (FP) are the data points that were misclassified as positive [Branco et al., 2016].

		Predicted	
		Positive	Negative
True	Positive	TP	FN
	Negative	FP	TN

Figure 3.9 *Confusion matrix for a binary classification problem [Branco et al., 2016].*

Some of the common metrics associated with a confusion matrix are:

- True Positive Rate (TPR): Also known as recall or sensitivity, it measures how well a classifier correctly predicts the positive class. It is computed as:

$$TPR = \frac{TP}{TP + FN}. \tag{3.7}$$

- True Negative Rate (TNR): Also known as specificity, this metric quantifies a classifier's ability to correctly predict the negative class. It is calculated as:

$$TNR = \frac{TN}{TN + FP}. \tag{3.8}$$

- Positive Predictive Value (PPV): The positive predictive value, also called precision, is the likelihood of a classifier not predicting as positive an observation that belongs to the negative class. The metric is given as follows:

$$PPV = \frac{TP}{TP + FP}. \tag{3.9}$$

- Negative Predictive Value (NPV): The negative predictive value is the likelihood of a classifier not predicting as negative an observation that belongs to the positive class. It is defined as follows:

$$NPV = \frac{TN}{TN + FN}. \tag{3.10}$$

- False Positive Rate (FPR): The false positive rate is the proportion of the negative class samples that were misclassified as positive. It is given as:

$$FPR = \frac{FP}{TN + FP} = 1 - TNR. \tag{3.11}$$

3.5.2 Accuracy

This is a simple and fundamental metric used for evaluating the performance of classification models. It measures how well a classifier correctly predicts the class of an instance by computing the ratio of correctly predicted instances to the total number of instances. For a binary classification task, accuracy is given by:

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}. \tag{3.12}$$

Although easy to compute and explain, accuracy can be misleading when there is an imbalanced class distribution. This is because it presumes a balanced class distribution, which is untrue in most situations. Consider, for example, an outlier detection problem where only 5% of the data belongs to the minority class (outliers) and 95% belongs to the majority class (inliers). A high accuracy of 95% is possible if the outlier detection model identifies every data point to be an inlier. This sounds interesting, however, it is deceiving because none of the outliers were identified.

3.5.3 Balanced Accuracy

Unlike accuracy, the balanced accuracy takes into consideration the predictive performance of a classifier on each class to produce a more reliable metric for tasks involving an imbalanced class distribution. For a binary classification problem, balanced accuracy is simply the average of TPR (sensitivity) and TNR (specificity) or the area under the receiver operating characteristic (ROC) curve with binary prediction labels instead of scores [Pedregosa et al., 2011]. It is computed as follows:

$$Balanced\ accuracy = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right). \tag{3.13}$$

3.5.4 Receiver Operating Characteristic

A receiver operating characteristic curve, also known as an ROC curve, is a popular diagnostic tool used in imbalanced class domains for evaluating classifier models. It is the most common evaluation metric for unsupervised outlier detection algorithms [Campos et al., 2016]. An ROC curve summarizes the behavior of a classifier by plotting the true positive rate (TPR) against the false positive rate (FPR) under various decision thresholds. Each point on the curve represents the performance of a model for a given threshold and class distribution. An ROC curve relies on a model to output a score or probability, which represents the degree to which a data point belongs to a particular class. For example, an outlier detection algorithm outputs a score to represent the degree of outlierness of an instance. Based on the predicted score, different thresholds are defined to generate the points on the ROC curve.

In Figure 3.10 below, are the ROC curves of three classifiers [Branco et al., 2016]. The point (0, 1) in the upper left corner of the curve represents a perfect classifier. A classifier model with an ROC point on the diagonal line is a random classifier, as it only makes a random guess of the class label. Any classifier, such as A and B in the figure, that is above the diagonal performs better than the random classifier. Conversely, a model that is below the diagonal line has a worse performance than the random classifier.

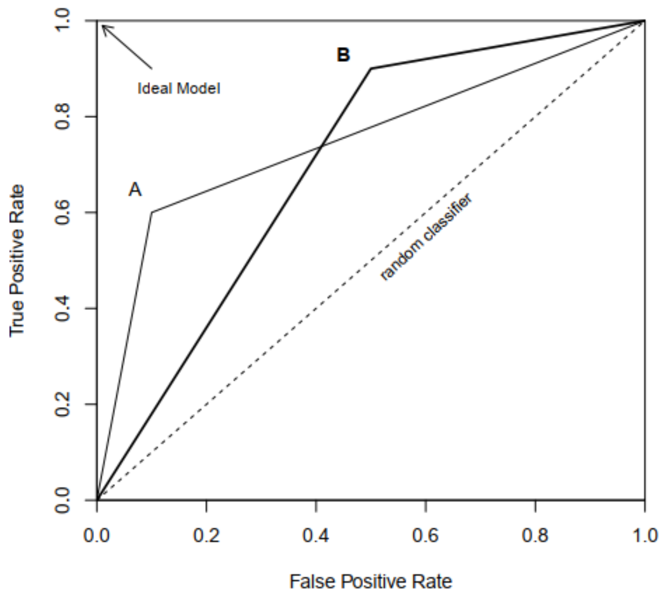

Figure 3.10 *Generated ROC curves of three classifiers [Branco et al., 2016].*

Different classifiers can be compared by computing the area under the receiver operating characteristic curve (ROC-AUC) to summarize their performance. The ROC-AUC is a single value between 0 and 1, as it represents a segment of the area of a unit square [Fawcett, 2006]. The greater the value of the ROC-AUC, the better the classifier's performance. Randomly guessing the data points, which corresponds to the diagonal on the ROC curve, would result in an ROC-AUC value of 0.5. However, an ROC-AUC value of 1 represents a perfect classification of the data instances.

3.6 Implementation

The coding implementation for this thesis was done using the Python programming language. The following open-source libraries were utilized for various aspects of the coding implementation.

- Matplotlib: Matplotlib is a library for the Python programming language used for creating data visualizations.
- NumPy: This is a fundamental library that provides numerical computing with Python.
- Pandas: Pandas is a Python library used for data manipulation and analysis.
- Scikit-learn: This is a machine learning library that provides tools for implementing supervised and unsupervised algorithms in Python. It also offers additional tools for data preprocessing, model evaluation, model selection, and model fitting.

4 Results and Discussions

This chapter presents the results of the evaluation metrics used to measure and compare the performance of the outlier detection algorithms. The chapter also details how these results were obtained and how the hyperparameters of the outlier detection models were optimized to achieve optimal model performance.

4.1 Preliminaries

The following points should be noted before presenting the results of the evaluation metrics.

- Data points with a target variable value of 0 represent inliers (positive class).
- Data points with a target variable value of 1 represent outliers (negative class).
- The fractions of outliers in the heart failure clinical records dataset and the epileptic seizure recognition dataset were 0.32 and 0.2, respectively.
- LOF, iForest, One-Class SVM, and MCD output both outlier scores and labels for each data point. These models output -1 for outliers and 1 for inliers, based on the outlier scores.
- DBSCAN only outputs cluster labels for each data point. It outputs -1 for outliers (noisy points).

4.2 Hyperparameter Optimization

Hyperparameters are parameters that cannot be directly estimated during the training process of machine learning models. The hyperparameters of the outlier detection algorithms must be properly tuned to identify and select the optimal ones that will be used in the actual outlier detection process. This will ensure that the correct hyperparameter values are chosen for the models, preventing overfitting and achieving optimal prediction performance. Each of the models was fitted with every possible combination of the hyperparameters by iterating over all possible value combinations. For every iteration, each of the feature scaling methods discussed in Section 3.3 of the previous chapter was used to transform the data before fitting the algorithm with the hyperparameter values. The datasets were rescaled to examine and understand the influence of scaling choice on the outlier detection models and to determine the optimal scaling for the datasets. Additionally, the algorithms were fitted with unscaled data during each iteration to evaluate the performance of the models on unscaled data.

For each of the scalers, the associated hyperparameter combination that produced the best evaluation result was identified. Thereafter, the hyperparameter combination, together with the associated scaler that produced the best overall evaluation result, was chosen as the final optimal hyperparameter combination. In the case of LOF, iForest, One-Class SVM, and MCD, the ROC-AUC score was used as the selection criterion. In contrast, the balanced accuracy score was used as the selection criterion for DBSCAN. The balanced accuracy score was used for DBSCAN because DBSCAN only predicts cluster labels, and prediction scores or probability estimates are required for computing the ROC-AUC score. Subsequently, to carry out outlier detection, the final optimal hyperparameter values were used to train each model with the associated scaler used to transform the data before model fitting. The following subsections give brief descriptions of the hyperparameters optimized for the outlier detection algorithms.

4.2.1 Hyperparameters in Local Outlier Factor

The hyperparameters tuned for this outlier detection algorithm are:

- n_neighbors (k): This is the number of neighbors to use in training the model [Pedregosa et al., 2011].
- p: This defines the distance metric to use [Pedregosa et al., 2011]. When $p = 1$, LOF uses Manhattan distance and Euclidean distance when $p = 2$.
- contamination: This refers to the fraction of outliers in the dataset. This value would always remain the same, as it is known from the datasets.

4.2.2 Hyperparameters in One-Class SVM

The following are the most important hyperparameters that were optimized for One-Class SVM.

- kernel: This specifies the type of kernel function to use in the algorithm. Commonly used kernel functions are linear, polynomial (poly), radial basis function (rbf), and sigmoid. The following equations represent the mathematical definitions of the kernel functions as given by [Pedregosa et al., 2011]. In these equations, r is specified by coef0, γ by gamma and d by degree.

$$linear: \langle x, x' \rangle \tag{4.1}$$

$$poly: (\gamma \langle x, x' \rangle + r)^d \tag{4.2}$$

$$rbf: exp(-\gamma \|x - x'\|^2) \tag{4.3}$$

$$sigmoid: tanh(\gamma \langle x, x' \rangle + r) \tag{4.4}$$

- gamma (γ): This is the kernel coefficient for rbf, poly, and sigmoid kernel types [Pedregosa et al., 2011]. In the scikit-learn implementation of One-Class SVM, gamma uses $1/(n\_features \times Var(X))$ and $1/n\_features$ when its value is given as *scale* and *auto*, respectively. In the equations, $Var(X)$ is the variance of the data, and $n\_features$ is the number of features in the dataset.
- coef0 (r): This is used to specify the independent term in poly and sigmoid kernel functions [Pedregosa et al., 2011]. It is not significant in linear and rbf kernels.
- nu (ν): This represents a lower bound for the fraction of support vectors and an upper bound for the fraction of training errors [Pedregosa et al., 2011]. It is used to regulate the fraction of outliers allowed, and as such, it would always be the same value as the fraction of outliers is known for both datasets.

4.2.3 Hyperparameters in Isolation Forest

The main hyperparameters tuned for iForest are:

- n_estimators (T): This is the number of base estimators in the ensemble. Base estimators are the trees in an ensemble [Pedregosa et al., 2011].
- max_samples (ψ): The number of data points to draw from the dataset to train each base estimator [Pedregosa et al., 2011].
- max_features: This represents the number of features to draw from the dataset to train each base estimator [Pedregosa et al., 2011].
- contamination: This refers to the fraction of outliers in the dataset. This value would always remain the same, as it is known from the datasets.

4.2.4 Hyperparameters in Minimum Covariance Determinant

In the case of the MCD algorithm, the two most important hyperparameters that can be tuned are:

- support_fraction (h): This is the fraction of data points to be included in the support of the raw MCD estimate [Pedregosa et al., 2011]. In the scikit-learn implementation of MCD, support_fraction uses $(n\_samples + n\_features + 1)/2 \times n\_samples$ when its value is given as *None*. The variables n_samples and n_features are, respectively, the number of observations and the number of features in the data.
- contamination: The fraction of outliers in the dataset, which would always be the same value.

4.2.5 Hyperparameters in DBSCAN

The hyperparameters eps and min_samples are very central to the performance of DBSCAN, and as such, they should be chosen appropriately.

- eps (ε): The eps defines the maximum distance between two data samples for them to be regarded as neighbors [Pedregosa et al., 2011]. In this thesis, the method described by [Ester et al., 1996] was used to choose the optimal values for eps. The basic idea of this approach consists of computing the k-th nearest neighbor distances for each point. These k-distances are sorted in ascending order and then plotted. The optimal value of eps corresponds to the value of k-distance at which a knee or elbow appears on the graph.
- min_samples ($MinPts$): It is the number of points within the eps neighborhood for a point to be considered a core point [Pedregosa et al., 2011]. This neighborhood also includes the point.

4.3 Outlier Detection in the Heart Failure Clinical Records Dataset

The heart failure clinical records dataset contains 96 outliers, accounting for approximately 32% (0.32) of the total data samples. To better visualize the data, the 12-dimensional feature space was projected into a 2D space using Principal Component Analysis (PCA). Figure 4.1 shows the plot of the first and second principal components from the PCA dimensionality reduction. In the plot, the red points represent the outliers, while the black points represent the inliers. A closer inspection of the plot reveals that the outliers are not easily distinguishable from the inliers, particularly in the lower part of the plot, where most of the data points are concentrated. Additionally, it was found that the first principal component accounts for 99.99% of the total variance in the data, indicating that most of the information in the data is captured in this component. The details of the hyperparameter optimization process for the heart failure clinical records dataset are presented in the following subsections.

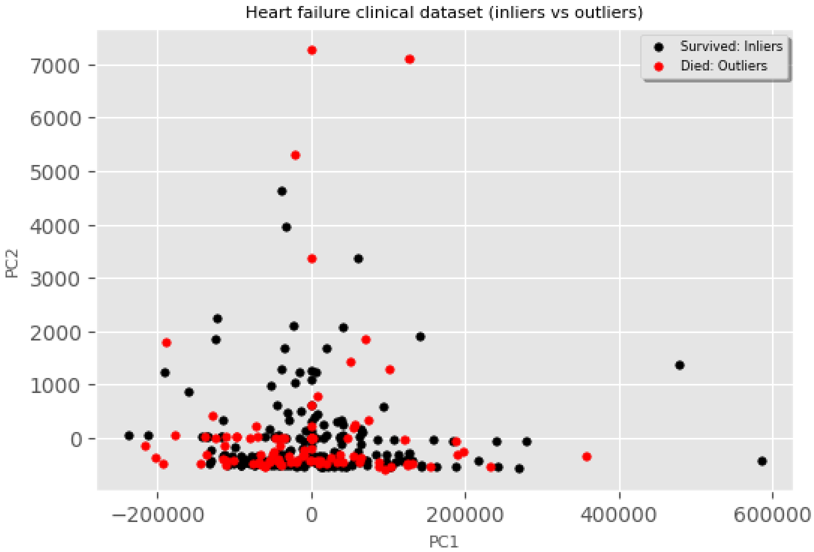

Figure 4.1 *PCA scatter plot of the heart failure clinical records dataset.*

4.3.1 Hyperparameter Optimization of Local Outlier Factor

The values considered in the hyperparameter optimization of the LOF algorithm are:

- n_neighbors: 20, 29, 59, 96, 104, 134, 149.
- p: 1, 2.
- contamination: 0.32.

Table 4.1 below shows the evaluation results of the hyperparameter tuning of LOF. As seen from the table, the Spline transformer achieved the best ROC-AUC score, followed by the MinMax scaler. Additionally, fitting the model with the unscaled data performed slightly better than the Normalizer. The final optimal hyperparameter values chosen for LOF to detect outliers in the heart failure clinical records dataset are thus, $n\_neighbors = 20$ and $p = 1$, with the data rescaled using the Spline transformer. Consequently, of the 299 samples in the data, 96 samples were predicted as outliers while 203 samples were predicted as inliers. Figure 4.2 shows the inliers and the outliers identified by the optimal LOF model.

Scaling method	n_neighbors	p	ROC-AUC score
Spline transformer	20	1	0.7474
MinMax scaler	29	1	0.7285
Standard scaler	59	1	0.7276
Robust scaler	149	1	0.7254
Unscaled data	134	1	0.5645
Normalizer	20	2	0.5622

Table 4.1 *LOF hyperparameter tuning evaluation results for the heart failure clinical records dataset.*

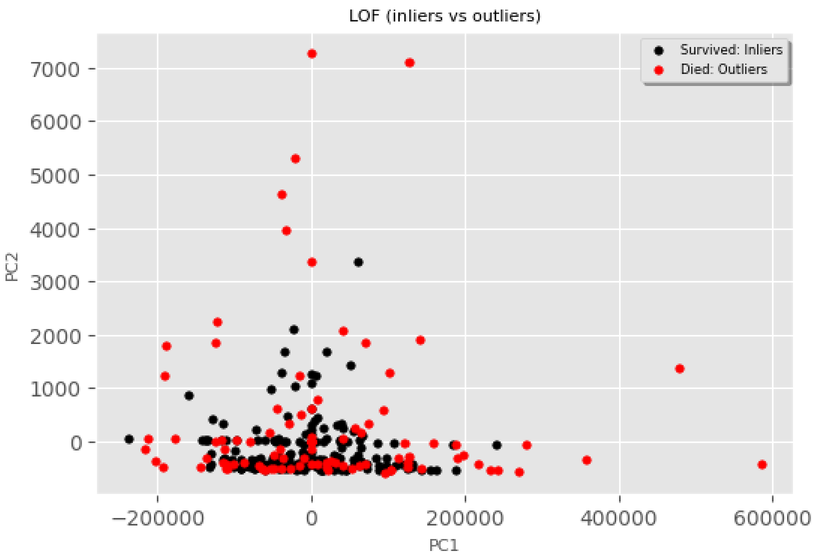

Figure 4.2 *LOF predicted inliers and outliers in the heart failure clinical records dataset.*

4.3.2 Hyperparameter Optimization of One-Class SVM

The following hyperparameter values were considered in the hyperparameter optimization of the One-Class SVM algorithm.

- kernel: poly, rbf, and sigmoid.
- gamma: scale, auto, 2.0, 1.75, 1.55, 1.5, 1.25, 1.05, 1.1, 1.0, 0.1, 0.01.
- coef0: 0.0, 0.1, 0.5, 1.0, 1.5.
- nu: 0.32.

The evaluation results obtained from the hyperparameter optimization of One-Class SVM are shown in Table 4.2 below. The ROC-AUC scores from the table indicate that the Normalizer performed worse than a random classifier. One explanation for this is that, unlike the other scaling techniques, Normalizer rescales each sample rather than each feature to have a unit norm, making the data highly susceptible to outliers. It can also be seen that the MinMax scaler attained the best score, followed closely by the Standard scaler. Hence, the final optimal hyperparameter values selected for One-Class SVM to detect outliers in the heart failure clinical records dataset are $kernel = rbf$ and $gamma = 1.5$, with the MinMax scaler used to scale the data. Finally, 95 data points were identified as outliers, while 204 data points were identified as inliers by the optimal One-Class SVM model. Figure 4.3 shows the predicted inliers and outliers.

Scaling method	kernel	gamma	Coef0	ROC-AUC score
MinMax scaler	rbf	1.5	N/A	0.6841
Standard scaler	rbf	0.1	N/A	0.6788
Spline transformer	rbf	1.1	N/A	0.6668
Unscaled data	rbf	2.0	N/A	0.6397
Robust scaler	rbf	0.1	N/A	0.6384
Normalizer	poly	2.0	1.5	0.4780

Table 4.2 *One-Class SVM hyperparameter tuning evaluation results for the heart failure clinical records dataset.*

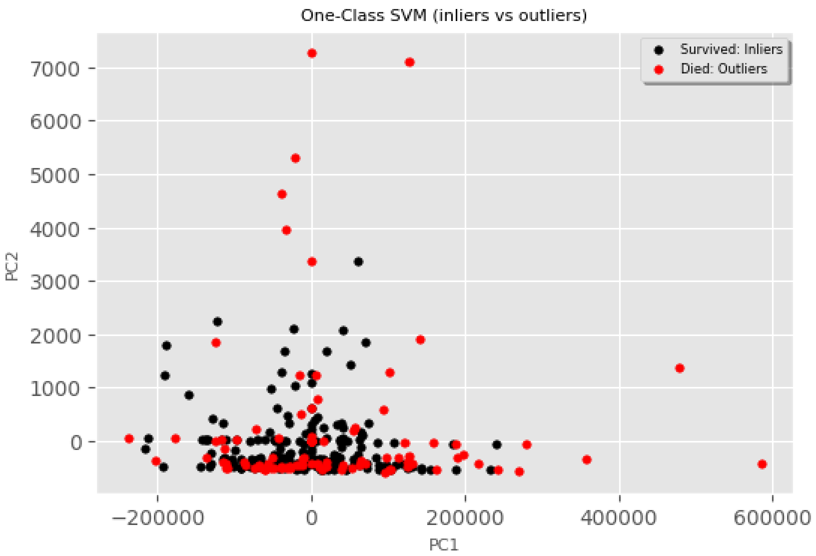

Figure 4.3 *One-Class SVM predicted inliers and outliers in the heart failure clinical records dataset.*

4.3.3 Hyperparameter Optimization of Isolation Forest

The values examined in the hyperparameter optimization of the iForest algorithm are:

- n_estimators: 100, 150, 200, 250, 300, 350, 400.
- max_samples: 179, 194, 209, 239, 254, 256, 269, 284, 299.
- max_features: 0.6, 0.8, 0.9, 1.0.
- contamination: 0.32.

Table 4.3 provides the ROC-AUC scores obtained from the hyperparameter optimization of iForest. It is apparent from this table that the dataset does not need to be scaled, which suggests that iForest, like other tree-based models, is relatively insensitive to scaling. It can also be seen from the table that the best performers only require 100 trees for each ensemble, as well as fractions of the number of samples and features. Therefore, the final optimal hyperparameter values chosen for iForest to detect outliers in the heart failure clinical records dataset are $n\_estimators = 100$, $max\_samples = 194$, and $max\_features = 0.9$, without rescaling the data, since the unscaled data is one of the best performers. As a result, out of the 299 observations in the dataset, 96 were classified as outliers while 203 were classified as inliers by the optimal iForest algorithm. The plot in Figure 4.4 shows the predicted inliers and outliers.

Scaling method	n_estimators	max_samples	max_features	ROC-AUC score
Unscaled data	100	194	0.9	0.7345
Robust scaler	100	194	0.9	0.7345
MinMax scaler	100	194	0.9	0.7345
Standard scaler	100	194	0.9	0.7345
Spline transformer	150	179	0.6	0.7310
Normalizer	150	269	1.0	0.6480

Table 4.3 *iForest hyperparameter tuning evaluation results for the heart failure clinical records dataset.*

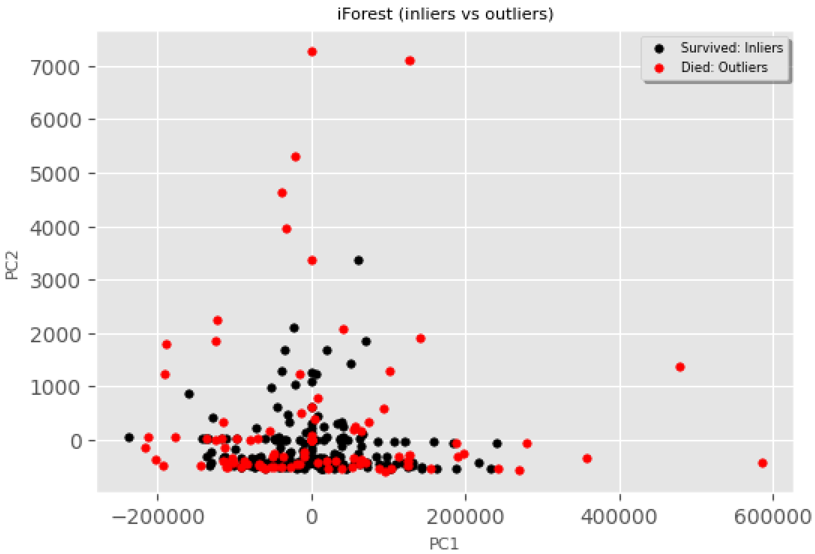

Figure 4.4 *iForest predicted inliers and outliers in the heart failure clinical records dataset.*

4.3.4 Hyperparameter Optimization of Minimum Covariance Determinant

The following values were used in the hyperparameter optimization process of the MCD algorithm.

- support_fraction: None, 0.6, 0.65, 0.7, 0.725, 0.75, 0.8, 0.825, 0.85, 0.875, 0.9.
- contamination: 0.32.

Table 4.4 below shows the evaluation results obtained from the hyperparameter optimization of MCD. As seen from the table, the Spline transformer achieved the best score. Therefore, 0.825 was chosen as the final optimal value for support_fraction to detect outliers in the heart failure clinical records dataset. Consequently, out of the 299 samples in the dataset, 96 samples were identified as outliers while 203 samples were identified as inliers by the optimal MCD model. The predicted inliers and outliers are presented in Figure 4.5.

Scaling method	support_fraction	ROC-AUC score
Spline transformer	0.825	0.7068
Standard scaler	0.9	0.6810
Robust scaler	0.9	0.6810
MinMax scaler	0.9	0.6810
Unscaled data	0.9	0.6810
Normalizer	0.875	0.6153

Table 4.4 *MCD hyperparameter tuning evaluation results for the heart failure clinical records dataset.*

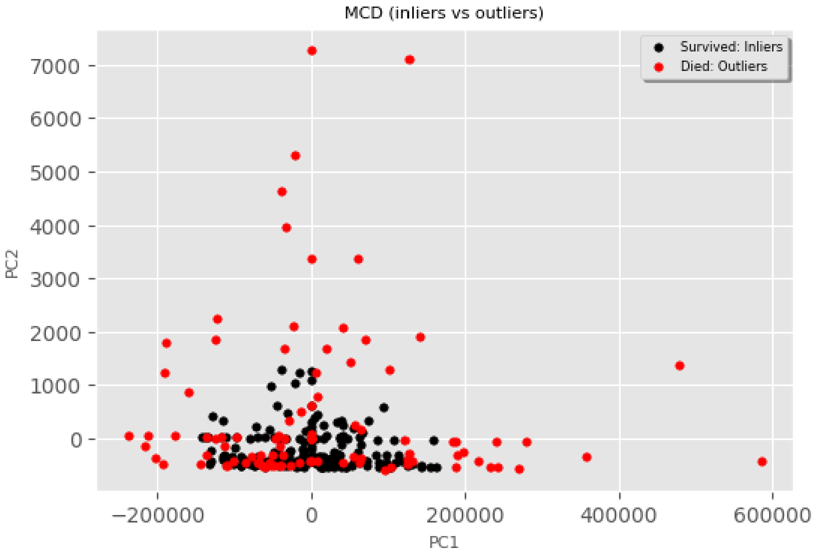

Figure 4.5 *MCD predicted inliers and outliers in the heart failure clinical records dataset.*

4.3.5 Hyperparameter Optimization of DBSCAN

To choose an appropriate range of values for eps, the k-th nearest neighbor distances for each point were computed after scaling the data with different preprocessing. These k-distances were then sorted and plotted in ascending order. The value of $k = 14$, which corresponds to the $MinPts$ (min_samples) in the DBSCAN algorithm, was used to specify the minimum number of points within the eps neighborhood. The resulting k-distance graphs are shown in Figure 4.6. As shown in the figure, the optimal eps value for each preprocessing corresponds to the point where the 14-NN distance curve exhibits an elbow. For example, the optimal eps value for the Spline transformer is around 1.5. Interestingly, there is no optimal eps value for Normalizer, as its plot remains a straight line. The following are the values of eps and min_samples optimized for the DBSCAN algorithm.

- eps: Based on the k-distance graphs in Figure 4.6, the range for eps was chosen in such a way that it includes all the realistic optimal eps values obtained for the various scaling methods. Thus, the values considered in the optimization of eps are 0.8, 0.9, 1.0, …, 3.2.
- min_samples: The values considered in tuning min_samples are 13, 14, 15, …, 26.

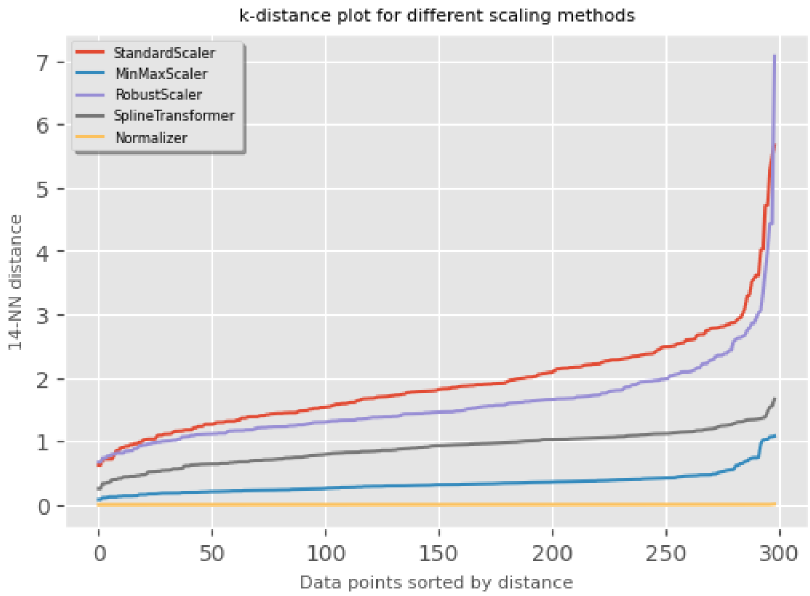

Figure 4.6 *Heart failure clinical records dataset k-distance graphs for eps.*

The evaluation results in Table 4.5 were obtained for the hyperparameter optimization of DBSCAN. The results for Normalizer and the unscaled data are missing because their trained models either did not predict more than one cluster label or detect any outliers. The table also shows that the best result was obtained with the Robust scaler. Therefore, $eps = 1.8$ and $min\_sample = 14$ were selected as the optimal hyperparameter values for DBSCAN to detect outliers in the heart failure clinical records dataset. As a result, of the 299 samples in the dataset, 151 samples were predicted as outliers while 148 samples were predicted as inliers by the optimal DBSCAN algorithm. Figure 4.7 shows the predicted inliers and outliers.

Scaling method	eps	min_samples	Balanced accuracy score
Robust scaler	1.8	14	0.6958
Standard scaler	2.7	19	0.6473
Spline transformer	1.3	17	0.6368
MinMax scaler	0.8	13	0.5579

Table 4.5 *DBSCAN hyperparameter tuning evaluation results for the heart failure clinical records dataset.*

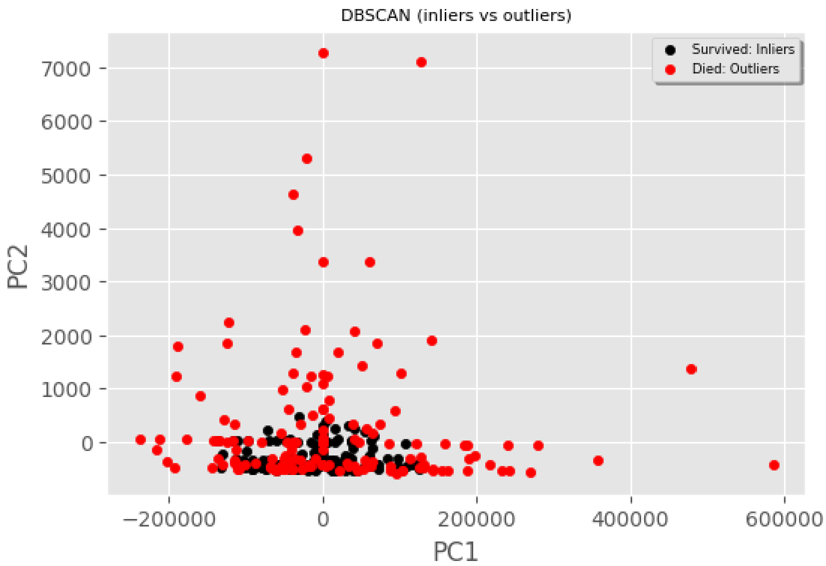

Figure 4.7 *DBSCAN predicted inliers and outliers in the heart failure clinical records dataset.*

4.3.6 Performance Evaluation for Heart Failure Clinical Records Dataset

The performance of the optimal trained algorithms used to detect outliers in the heart failure clinical records dataset was analyzed and compared. First, the performance of LOF, One-Class SVM, iForest, and MCD was assessed and compared using their ROC-AUC scores. Figure 4.8 shows the ROC curves generated for these models. The LOF outperformed the other algorithms in terms of AUC score. Interestingly, MCD performed better than One-Class SVM.

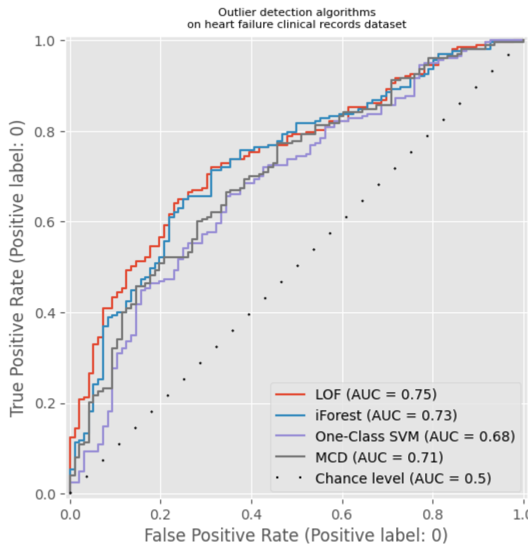

Figure 4.8 *ROC curves for the optimal trained models used to detect outliers in the heart failure clinical records dataset.*

Next, the metric results obtained from the confusion matrices of the optimal algorithms were evaluated and compared. Table 4.6 presents the results of the different metrics. As can be seen from the table, DBSCAN outperformed other algorithms in correctly detecting outliers

($TN = 74$) and in wrongly predicting outliers as inliers ($FP = 22$). However, it performed worse than the remaining models in accurately classifying inliers ($TP = 126$), and in falsely predicting inliers as outliers ($FN = 77$), raising more false alarms than the other models. Among iForest, LOF, MCD, and One-Class SVM, iForest has the highest number of accurately classified outliers and inliers, and the fewest number of misclassified outliers and inliers.

Model	TP	FP	FN	TN
DBSCAN	126	22	77	74
iForest	158	45	45	51
LOF	157	46	46	50
MCD	157	46	46	50
One-Class SVM	153	51	50	45

Table 4.6 *Metric results from confusion matrices for the heart failure clinical records dataset.*

Furthermore, Table 4.7 presents the results derived from the metrics of the confusion matrices for the optimal outlier detection models. It can be seen from the table that DBSCAN has the highest TNR value, implying that it is more likely for DBSCAN to correctly predict an outlier in the data than any of the other models. DBSCAN also has the highest PPV value, meaning it has the highest probability of not predicting an outlier as an inlier. However, DBSCAN has the lowest TPR value, indicating that it is less efficient in accurately predicting inliers. In terms of NPV, DBSCAN only outperformed One-Class SVM. Among iForest, LOF, MCD, and One-Class SVM, iForest has the highest TPR, TNR, PPV, and NPV values.

Model	TPR	TNR	PPV	NPV
DBSCAN	0.6207	0.7708	0.8514	0.4901
iForest	0.7783	0.5312	0.7783	0.5312
LOF	0.7734	0.5208	0.7734	0.5208
MCD	0.7734	0.5208	0.7734	0.5208
One-Class SVM	0.7537	0.4688	0.75	0.4737

Table 4.7 *Derived metric results from confusion matrices for the heart failure clinical records dataset.*

Finally, another significant metric to consider is the balanced accuracy score, which measures the combined performance of the algorithms on both the majority class (inliers) and the minority class (outliers). Table 4.8 shows the balanced accuracy and the accuracy scores of the optimal outlier detection models. Although the accuracy scores can be deceptive in the context of imbalanced classes, they are presented in the table to illustrate their unreliability. In the table, DBSCAN has the highest balanced accuracy score, followed by iForest. Other algorithms, except One-Class SVM, have better accuracy scores than DBSCAN. This would have been misleading if only the accuracy scores were considered because accuracy only computes the ratio of correctly predicted data points to the total number of observations, thereby favoring the majority class (inliers).

Model	Balanced accuracy score	Accuracy score
DBSCAN	69.58%	66.89%
iForest	65.48%	69.9%
LOF	64.71%	69.23%
MCD	64.71%	69.23%
One-Class SVM	61.12%	66.22%

Table 4.8 *Balanced accuracy and accuracy scores for the heart failure clinical records dataset.*

4.4 Outlier Detection in the Epileptic Seizure Recognition Dataset

The epileptic seizure recognition dataset contains 2300 outliers, accounting for 20% (0.2) of the total observations. To better visualize the data, the 178-dimensional feature space was projected into a 2D space using PCA. Figure 4.9 shows the plot of the first and second principal components from the PCA dimensionality reduction. In the plot, the red points represent the outliers, while the black points represent the inliers. The outliers are largely separated from the inliers. Additionally, the first and second principal components account for 10.84% of the total variance in the data, therefore, a significant amount of variance remains in the remaining principal components. The details of the hyperparameter optimization process for the epileptic seizure recognition dataset are presented in the following subsections.

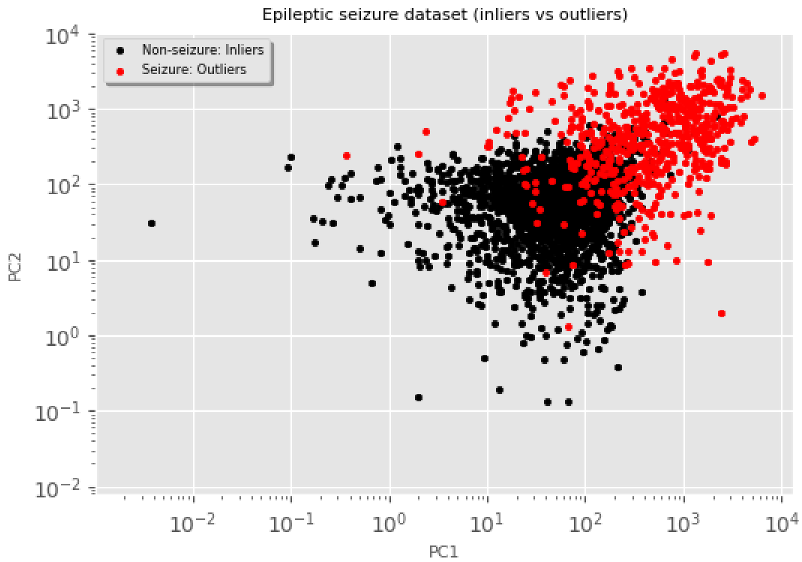

Figure 4.9 *PCA scatter plot of the epileptic seizure recognition dataset.*

4.4.1 Hyperparameter Optimization of Local Outlier Factor

The following values were used in the hyperparameter optimization of the LOF algorithm.

- n_neighbors: 115, 230, 345, 460, 575.
- p: 1, 2.
- contamination: 0.2.

The evaluation results obtained from the hyperparameter tuning of LOF are presented in Table 4.9. It can be seen from the table that all the scalers except the Normalizer performed exceedingly well, almost achieving a perfect ROC-AUC score. The marginal differences in their scores suggest that any of these scalers would be an ideal candidate for rescaling the data. It is also worth noting that training the model on the unscaled data also produced an almost perfect score. Furthermore, the best score was obtained with the Standard scaler, followed by the unscaled data. Thus, the final optimal hyperparameter values selected for LOF to detect outliers in the epileptic seizure recognition dataset are $n\_neighbors = 230$ and $p = 2$, with the Standard scaler used to preprocess the data. As a result, out of the 11500 samples in the data, 2300 samples were predicted as outliers, while 9200 samples were predicted as inliers by the optimal LOF model. The predicted inliers and outliers are shown in Figure 4.10.

Scaling method	n_neighbors	p	ROC-AUC score
Standard scaler	230	2	0.992490
Unscaled data	230	2	0.992472
Robust scaler	230	2	0.992469
MinMax scaler	230	2	0.992466
Spline transformer	230	2	0.992091
Normalizer	575	1	0.560427

Table 4.9 *LOF hyperparameter tuning evaluation results for the epileptic seizure recognition dataset.*

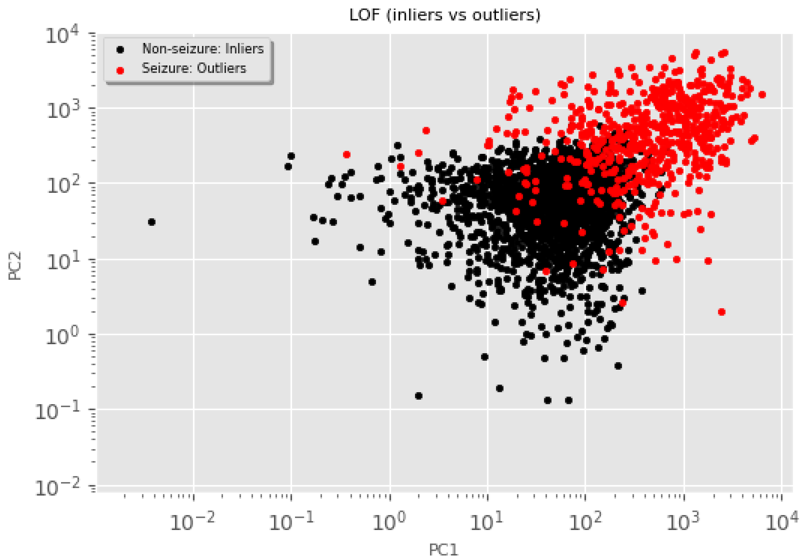

Figure 4.10 *LOF predicted inliers and outliers in the epileptic seizure recognition dataset.*

4.4.2 Hyperparameter Optimization of One-Class SVM

The values considered in the hyperparameter optimization process of the One-Class SVM algorithm are:

- kernel: rbf.
- gamma: scale, auto, 2.0, 1.75, 1.55, 1.5, 1.25, 1.05, 1.1, 1.0, 0.1, 0.01.
- nu: 0.2.

The results of the hyperparameter optimization of One-Class SVM are shown in Table 4.10. Here, only rbf kernel was considered because of its flexibility and effectiveness in capturing complex relationships in data. As shown in the table, the best score was achieved with the Spline transformer, followed by the Robust scaler. Hence, to detect outliers in the epileptic seizure recognition dataset, 1.25 was picked as the final optimal value for gamma. Of the 11500 data points in the data, 2313 were detected as outliers, while 9187 were identified as inliers by the optimal One-Class SVM algorithm. Figure 4.11 shows the predicted inliers and outliers.

Scaling method	kernel	gamma	ROC-AUC score
Spline transformer	rbf	1.25	0.990722
Robust scaler	rbf	auto	0.990246
Standard scaler	rbf	0.01	0.989607
MinMax scaler	rbf	auto	0.989246
Unscaled data	rbf	scale	0.989210
Normalizer	rbf	0.01	0.756366

Table 4.10 *One-Class SVM hyperparameter tuning evaluation results for the epileptic seizure recognition dataset.*

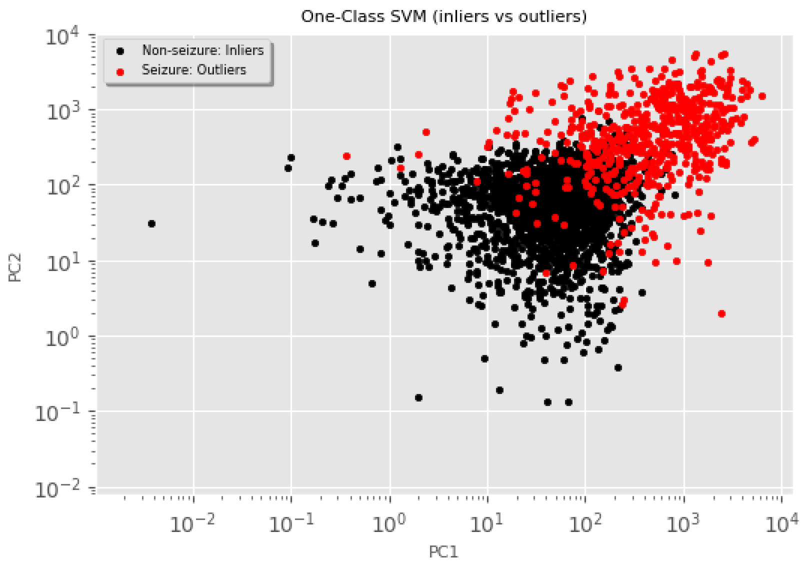

Figure 4.11 *One-Class SVM predicted inliers and outliers in the epileptic seizure recognition dataset.*

4.4.3 Hyperparameter Optimization of Isolation Forest

The following values were used in the hyperparameter optimization of the iForest algorithm.

- n_estimators: 100, 150, 200, 250, 300, 350, 400.
- max_samples: 256, 345, 460, 575, 690, 805.
- max_features: 0.6, 0.8, 0.9, 1.0.
- contamination: 0.2.

Table 4.11 below presents the evaluation results obtained from the hyperparameter tuning of iForest. It can be seen from the table that the best score was obtained with the Spline transformer, followed by the Standard scaler. Therefore, $n\_estimators = 400$, $max\_samples = 256$ and $max\_features = 0.8$ were chosen as the hyperparameter values for iForest to detect the outliers in the epileptic seizure recognition dataset. Of the 11500 observations in the dataset, the optimal iForest model classified 2300 data points as outliers and 9200 data points as inliers. The predicted inliers and outliers are shown in Figure 4.12.

Scaling method	n_estimators	max_samples	max_features	ROC-AUC score
Spline transformer	400	256	0.8	0.989923
Standard scaler	350	345	1.0	0.989744
Robust scaler	350	345	1.0	0.989744
MinMax scaler	350	345	1.0	0.989744
Unscaled data	350	345	1.0	0.989744
Normalizer	400	0.06	0.9	0.662677

Table 4.11 *iForest hyperparameter tuning evaluation results for the epileptic seizure recognition dataset.*

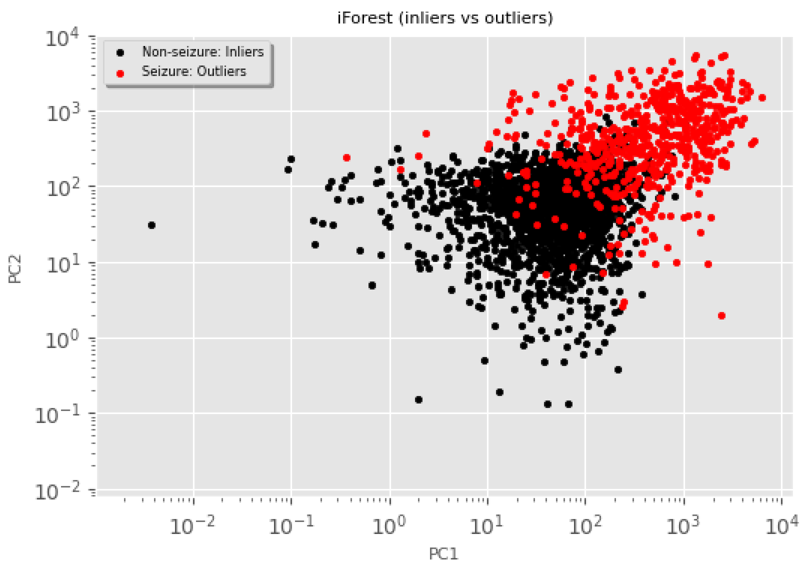

Figure 4.12 *iForest predicted inliers and outliers in the epileptic seizure recognition dataset.*

4.4.4 Hyperparameter Optimization of Minimum Covariance Determinant

The values considered in the hyperparameter optimization of the MCD algorithm are:

- support_fraction: None, 0.6, 0.65, 0.7, 0.725, 0.75, 0.8, 0.825, 0.85, 0.875, 0.9.
- contamination: 0.2.

The evaluation results obtained from the hyperparameter optimization of MCD are shown in Table 4.12. One notable observation from the table is that the Normalizer performed far worse than a random classifier. One reason for this is that, unlike the other scalers, the Normalizer scales each sample rather than each feature to a unit norm, thereby making the outliers undetectable by algorithms that expect consistency in the data transformation. It can also be seen from the table that the Spline transformer achieved the best score. Therefore, to detect outliers in the epileptic seizure recognition dataset, 0.875 was chosen as the final optimal value for support_fraction. As a result, of the 11500 samples in the data, 2300 samples were predicted as outliers, while 9200 samples were predicted as inliers by the optimal MCD model. Figure 4.13 shows the predicted inliers and outliers.

Scaling method	support_fraction	ROC-AUC score
Spline transformer	0.875	0.985778
Standard scaler	0.6	0.962098
Robust scaler	0.6	0.962098
MinMax scaler	0.6	0.962098
Unscaled data	0.6	0.962098
Normalizer	0.9	0.340150

Table 4.12 *MCD hyperparameter tuning evaluation results for the epileptic seizure recognition dataset.*

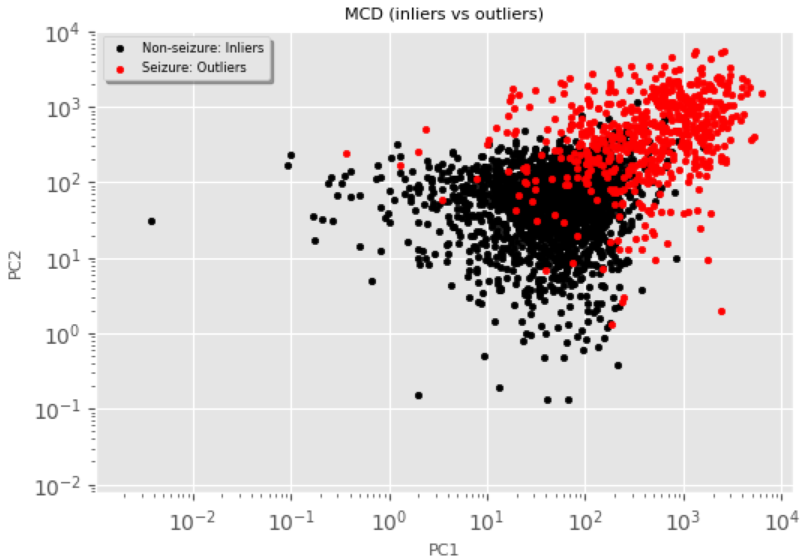

Figure 4.13 *MCD predicted inliers and outliers in the epileptic seizure recognition dataset.*

4.4.5 Hyperparameter Optimization of DBSCAN

Following the procedure described in Section 4.3.5, the k-distances for each sample in the epileptic seizure recognition dataset were calculated after scaling the data with different preprocessing. The value of $k = 326$ was used to specify the minimum number of points within the eps neighborhood. Figure 4.14 presents the resulting k-distance graphs. There are no optimal eps values for the MinMax scaler and Normalizer, as their plots remain almost a straight line. The following are the values of eps and min_samples optimized for the DBSCAN algorithm.

- eps: Based on the k-distance curves in Figure 4.14, the range for eps was chosen in such a way that it includes all the realistic optimal eps values obtained for the various scalers. Thus, the values considered in optimizing eps are 5.0, 7.5, 10.0, 12.5, 15.0, 17.5, 20.0.

- min_samples: The values considered in optimizing min_samples are 326, 336, 346, 356, 366.

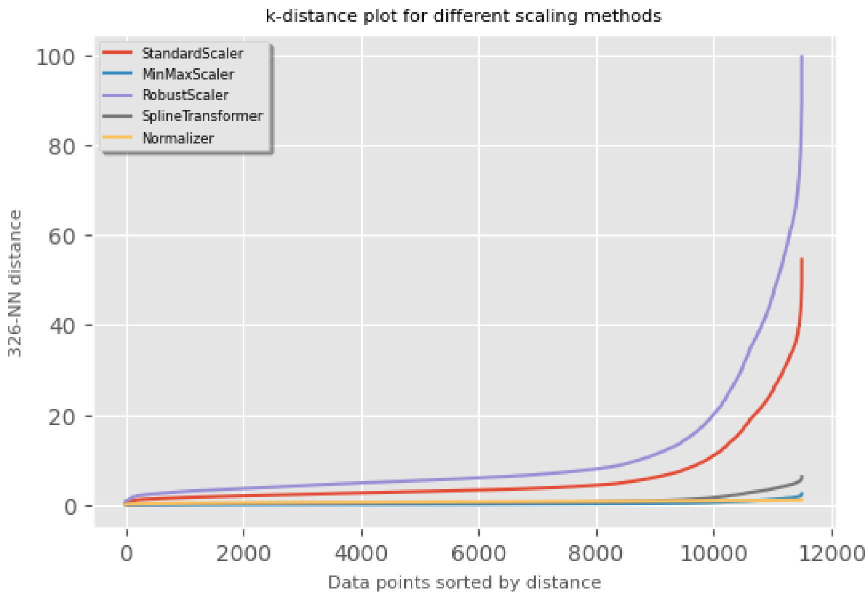

Figure 4.14 *Epileptic seizure recognition dataset k-distance graphs for eps.*

The evaluation results obtained from the hyperparameter optimization of the DBSCAN algorithm are presented in Table 4.13. The results for the MinMax scaler, the Normalizer, and the unscaled data are missing because their trained models did not predict more than one cluster label or detect any outliers. In the table, the best result was achieved with the Robust scaler, followed by the Standard scaler. Therefore, to detect outliers in the epileptic seizure recognition dataset, $eps = 12.5$ and $min\_samples = 326$ were picked as the final optimal hyperparameter values for DBSCAN, with the Robust scaler used to rescale the dataset. Out of the 11500 samples in the dataset, 2491 samples were classified as outliers, while 9009 samples were classified as inliers by the optimal DBSCAN model. The predicted inliers and outliers are shown in Figure 4.15.

Scaling method	eps	min_samples	Balanced accuracy score
Robust scaler	12.5	326	0.9611
Standard scaler	7.5	326	0.9483
Spline transformer	5	326	0.5278

Table 4.13 *DBSCAN hyperparameter tuning evaluation results for the epileptic seizure recognition dataset.*

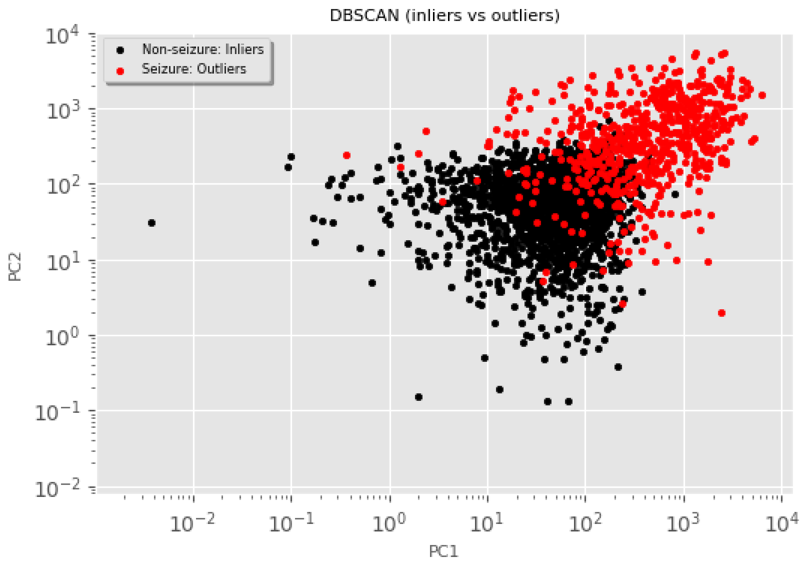

***Figure 4.15** DBSCAN predicted inliers and outliers in the epileptic seizure recognition dataset.*

4.4.6 Performance Evaluation for Epileptic Seizure Recognition Dataset

The performance of the optimal trained models used to detect outliers in the epileptic seizure recognition dataset was evaluated and compared. To begin, the performance of LOF, One-Class SVM, iForest, and MCD was compared using their ROC-AUC scores. As shown in Figure 4.16, all four models achieved a score of 0.99, indicating an almost perfect performance.

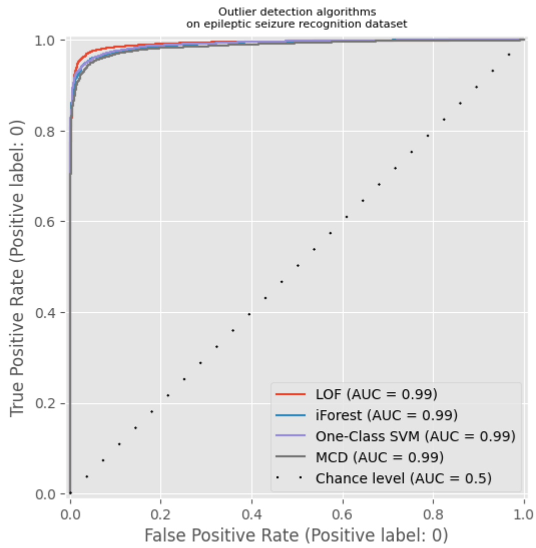

***Figure 4.16** ROC curves for the optimal trained models used to detect outliers in the epileptic seizure recognition dataset.*

Next, the metric results from the confusion matrices of the optimal models were analyzed and examined. The results are presented in Table 4.14. From the results in the table, the DBSCAN algorithm performed better than the other algorithms in accurately detecting outliers ($TN = 2195$) and in misclassifying outliers as inliers ($FP = 105$). However, it performed worse than the other algorithms in correctly identifying inliers ($TP = 8904$) and in wrongly predicting inliers as outliers ($FN = 296$), raising more false alarms than the other models. Among iForest, LOF, MCD, and One-Class SVM, LOF has the highest number of correctly predicted outliers and inliers, and the fewest number of incorrectly predicted outliers and inliers.

Model	TP	FP	FN	TN
DBSCAN	8904	105	296	2195
LOF	9026	174	174	2126
One-Class SVM	8969	218	231	2082
iForest	8960	240	240	2060
MCD	8942	258	258	2042

Table 4.14 *Metric results from confusion matrices for the epileptic seizure recognition dataset.*

To gain further insights into the models' performance, the results in Table 4.15 were derived from the metrics of the models' confusion matrices. As shown in the table, DBSCAN achieved the highest TNR value, indicating that it is more likely for DBSCAN to accurately identify outliers in the data than the other algorithms. DBSCAN also has the highest PPV value, indicating that it has the highest probability of not predicting an outlier as an inlier. On the other hand, DBSCAN has the lowest TPR value, suggesting that it is less effective in accurately identifying inliers. It is also worth mentioning that the DBSCAN algorithm has the lowest NPV value, signifying it is more likely for the DBSCAN model to misclassify an inlier. Among iForest, LOF, MCD, and One-Class SVM, LOF has the highest TPR, TNR, PPV, and NPV values.

Model	TPR	TNR	PPV	NPV
DBSCAN	0.9678	0.9543	0.9883	0.8812
LOF	0.9811	0.9243	0.9811	0.9243
One-Class SVM	0.9749	0.9052	0.9763	0.9001
iForest	0.9739	0.8957	0.9739	0.8957
MCD	0.9720	0.8878	0.9720	0.8878

Table 4.15 *Derived metric results from confusion matrices for the epileptic seizure recognition dataset.*

Lastly, Table 4.16 shows the balanced accuracy and the accuracy scores of the optimal outlier detection algorithms. It is apparent from the table that DBSCAN has the highest balanced accuracy score, followed by LOF. This indicates that DBSCAN has a better combined performance than the remaining algorithms on both the inliers and the outliers. Conversely, LOF has the highest accuracy score, followed by DBSCAN. This is deceiving as accuracy only favors the majority class (inliers) in its computation.

Model	Balanced accuracy score	Accuracy score
DBSCAN	96.11%	96.51%
LOF	95.27%	96.97%
One-Class SVM	94%	96.1%
iForest	93.48%	95.83%
MCD	92.99%	95.51%

Table 4.16 *Balanced accuracy and accuracy scores for the epileptic seizure recognition dataset.*

5 Conclusions

This thesis aimed to utilize unsupervised outlier detection algorithms to identify outliers in two publicly available datasets and to evaluate and compare their performance. The outliers in the datasets represent the negative class, while the inliers belong to the positive class. The optimal hyperparameter values of the models and the appropriate scaling methods for the datasets were selected based on the results obtained from the hyperparameter optimization process. One significant finding from the optimization process is that the Normalizer produced the worst results when used to transform the datasets. Thus, it was never selected as an appropriate scaler for any of the datasets. In some cases, the Normalizer performed worse than a random classifier, and when used with DBSCAN, no outlier or only one cluster was discovered. This is because the Normalizer, unlike the other scalers, rescales each sample vector instead of each feature to a unit norm, thereby making the data highly prone to outliers. Another notable discovery from the optimization process confirmed that Isolation Forest, just like other tree-based algorithms, is hardly affected by scaling.

The results from the performance analysis and comparison of the outlier detection algorithms show that DBSCAN demonstrated superior performance in accurately detecting outliers in the datasets. DBSCAN achieved the highest number of true negatives, correctly detecting approximately 77.1% and 95.4% of the outliers in the heart failure clinical records dataset and the epileptic seizure recognition dataset, respectively. However, it achieved the lowest number of true positives among the algorithms, accurately identifying approximately 62.1% and 96.8% of the inliers in the heart failure clinical records dataset and the epileptic seizure recognition dataset, respectively. As a result, the DBSCAN algorithm raised more false alarms (false negatives) than the other algorithms.

The results of this thesis have confirmed that DBSCAN is an efficient algorithm for detecting outliers in data when its hyperparameters are thoroughly and adequately tuned. Its proficiency in detecting outliers is attributed to its ability to discover arbitrary-shaped clusters or clusters of varying densities. However, as shown in this thesis, the high number of false alarms can reduce the reliability of the algorithm. These wrongly classified data points, which, according to DBSCAN, belong to the sparse clusters, are not outliers but rather inliers.

Further work is needed to validate the outliers and inliers detected by unsupervised outlier detection models, thereby improving their credibility. To reduce the number of false alarms, Ahmed et al. [2018] proposed an outlier detection system that utilizes a clustering algorithm (DBSCAN) to identify outliers and a classifier (SVM) to validate the detected outliers. Also, Rashid et al. [2024] employed a hybrid model to detect intrusions in an intrusion detection system. The model utilizes *K*-Means clustering to partition the data into clusters, DBSCAN to detect outliers in the partitioned clusters, and Sequential Minimal Optimization (SMO) based SVM to validate the detected outliers.

References

Ahmad, Tanvir, Assia Munir, Sajjad Haider Bhatti, Muhammad Aftab, and Muhammad Ali Raza. 2017. Survival analysis of heart failure patients: a case study. *PLoS One* 12 (7), e0181001-e0181001.

Ahmed, Esraa Samir, Laila A. Abd-Elmegid, and Hala Abdel-Galil. 2018. Efficient model to reduce false positives using outliers detection in big data. *Communications on Applied Electronics* 7 (24), 11-15.

Amer, Mennatallah, Markus Goldstein, and Slim Abdennadher. 2013. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, 8-15.

American Heart Association. 2023. Ejection Fraction Heart Failure Measurement. Retrieved October 2024. *https://www.heart.org/en/health-topics/heart-failure/diagnosing-heart-failure/ejection-fraction-heart-failure-measurement*.

Andrzejak, Ralph G., Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E. Elger. 2001. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: dependence on recording region and brain state. *Physical Review E* 64(6), Article 061907, 8 pages.

Askarin, Mohammad Mogharen, Jiankun Hu, Min Wang, Xuefei Yin, and Xiuping Jia. 2025. A B-Spline function based 3D point cloud unwrapping scheme for 3D fingerprint recognition and identification. *IEEE Open Journal of the Computer Society* 6, 480-90.

Boukerche, Azzedine, Lining Zheng, and Omar Alfandi. 2020. Outlier detection: methods, models, and classification. *ACM Computing Surveys (CSUR)* 53 (3), 1-37.

Bounsiar, Abdenour and Michael G. Madden. 2014. One-class support vector machines revisited. *2014 International Conference on Information Science & Applications (ICISA)*, 1-4.

Branco, Paula, Luís Torgo, and Rita P. Ribeiro. 2016. A survey of predictive modeling on imbalanced domains. *ACM computing surveys* (CSUR) 49 (2), 1-50.

Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. *SIGMOD Record* 29 (2), 93–104.

Campos, Guilherme O., Arthur Zimek, Jörg Sander, Ricardo J. G. B. Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E. Houle. 2016. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery* 30 (4), 891-927.

Case-Lo, Christine. 2018. Blood Sodium Test. Health Line. Retrieved October 2024. *https://www.healthline.com/health/sodium-blood*.

Chandola, Varun, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: a survey. *ACM Computing Surveys* 41, 3, Article 15, 58 pages.

Chen, Michael A. 2023. Creatine phosphokinase test. MedlinePlus. Retrieved October 2024. *https://medlineplus.gov/ency/article/003503.htm*.

Chicco, Davide and Giuseppe Jurman. 2020. Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. *BMC Medical Informatics and Decision Making* 20 (1), 1-16.

Cleveland Clinic. 2022. Ejection Fraction. Retrieved October 2024. *https://my.clevelandclinic.org/health/articles/16950-ejection-fraction*.

Cleveland Clinic. 2022. Epilepsy. Retrieved October 2024. *https://my.clevelandclinic.org/health/diseases/17636-epilepsy*.

Demolder, Anthony. 2024. Left Ventricular Systolic Dysfunction: The Basics and AI Advancements in Detection. Powerful Medical. Retrieved November 2024. *https://www.powerfulmedical.com/blog/left-ventricular-systolic-dysfunction-lvsd/*.

Domingues, Rémi, Maurizio Filippone, Pietro Michiardi, and Jihane Zouaoui. 2018. A comparative evaluation of outlier detection algorithms: experiments and analyses. *Pattern Recognition* 74, 406-421.

Ester, Martin, Hans-Peter Kriegel, Jiirg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings - 2nd International Conference on Knowledge Discovery and Data Mining, KDD 1996*, 226-231.

Fawcett, Tom. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27 (8), 861-874.

Filzmoser, Peter, Robert G. Garrett, and Clemens Reimann. 2005. Multivariate outlier detection in exploration geochemistry. *Computers & Geosciences* 31 (5), 579-587.

Fritsch, Virgile, Gaël Varoquaux, Benjamin Thyreau, Jean-Baptiste Poline, and Bertrand Thirion. 2012. Detecting outliers in high-dimensional neuroimaging datasets with robust covariance estimators. *Medical Image Analysis* 16 (7), 1359-1370.

Goldstein, Markus and Seiichi Uchida. 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *Plus One* 11 (4), 31 pages.

Han, Jiawei, Jian Pei, and Micheline Kamber. 2012. *Data Mining: Concepts and Techniques*. 3rd ed., Morgan Kaufmann, 2012.

Hawkins, D. 1980. *Identification of Outliers.* 1st ed., Springer Netherlands, 1980.

Heart Failure Clinical Records. 2020. UCI Machine Learning Repository. *https://doi.org/10.24432/C5Z89R*.

Hettich, Seth and S. D. Bay. 1999. The UCI KDD archive [http://kdd. ics. uci. edu]. *Irvine, CA: University of California. Department of Information and Computer Science*.

Hubert, Mia, Michiel Debruyne, and Peter J. Rousseeuw. 2018. Minimum covariance determinant and extensions. *Wiley Interdisciplinary Reviews: Computational Statistics* 10 (3), e1421.

IBM. 2023. What is principal component analysis (PCA)? Retrieved July 2025. https://www.ibm.com/think/topics/principal-component-analysis.

Karpov, Oleg E., Matvey S. Khoymov, Vladimir A. Maksimenko, Vadim V. Grubov, Nikita Utyashev, Denis A. Andrikov, Semen A. Kurkin, and Alexander E. Hramov. 2023. Evaluation of unsupervised anomaly detection techniques in labelling epileptic seizures on human EEG. *Applied Sciences* 13 (9), 5655.

Khoirunnisa, Azka and Nur Ghaniaviyanto Ramadhan. 2023. Improving malaria prediction with ensemble learning and robust scaler: an integrated approach for enhanced accuracy. *Jurnal Infotel* 15 (4), 326–34.

Kode, Hepseeba, Khaled Elleithy, and Laiali Almazaydeh. 2024. Epileptic seizure detection in EEG signals using machine learning and deep learning techniques. *IEEE Access* 12, 80657-80668.

Kriegel, Hans-Peter, Kröger, Peer, and Arthur Zimek. 2010. Outlier detection techniques. *Tutorial at KDD*, 10, 1-76.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data* 6 (1), 1-39.

Liu, Yufang, Bin Jiang, and Hui Yi. 2024. Online detection of analog signal anomaly based on the evaluation of local trends. *Measurement Science and Technology* 35 (1), 015040.

Manevitz, Larry M. and Malik Yousef. 2001. One-class SVMs for document classification. *Journal of Machine Learning Research* 2, 139-154.

Mormann, Florian, Ralph G. Andrzejak, Christian E. Elger, and Klaus Lehnertz. 2007. Seizure prediction: the long and winding road. *Brain* 130 (2), 314-333.

Naser, Marwah Abdulrazzaq, Aso Ahmed Majeed, Muntadher Alsabah, Taha Raad Al-Shaikhli, and Kawa M. Kaky. 2024. A review of machine learning's role in cardiovascular disease prediction: recent advances and future challenges. *Algorithms* 17 (2), 78.

National Health Service (NHS). 2022. Cardiovascular disease. Retrieved November 2024. *https://www.nhs.uk/conditions/cardiovascular-disease/.*

National Heart, Lung, and Blood Institute. 2022. Platelet Disorders. Retrieved October 2024. *https://www.nhlbi.nih.gov/health/thrombocytopenia*.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research* 12, 2825-2830.

Prastawa, Marcel, Elizabeth Bullitt, Sean Ho, and Guido Gerig. 2004. A brain tumor segmentation framework based on outlier detection. *Medical Image Analysis* 8 (3), 275-283.

Rashid, Umair, Muhammad Faheem Saleem, Saad Rasool, Ahmad Abdullah, Hira Mustafa, and Aiman Iqbal. 2024. Anomaly detection using clustering (K-Means with DBSCAN) and SMO. *Journal of Computing & Biomedical Informatics* 7 (02).

Rousseeuw, Peter J. 1984. Least median of squares regression. *Journal of the American Statistical Association* 79 (388), 871-880.

Rousseeuw, Peter J. and Driessen, Katrien Van. 1999. A fast algorithm for the minimum covariance determinant estimator. *Technometrics* 41 (3), 212-223.

Sai, Chong Yeh, Norrima Mokhtar, Masahiro Iwahashi, Paul Cumming, and Hamzah Arof. 2021. Fully automated unsupervised artefact removal in multichannel electroencephalogram using wavelet-independent component analysis with density-based spatial clustering of application with noise. *IET Signal Processing* 15 (8), 535-542.

Schölkopf, Bernhard, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural Computation* 13 (7), 1443–1471.

Shin, Hyum Joon, Dong-Hwan Eom, and Sung-Shick Kim. 2005. One-class support vector machines—an application in machine fault detection and classification. *Computers & Industrial Engineering* 48 (2), 395-408.

Stephens, Carissa. 2023. Creatinine Blood Test. Health Line. Retrieved October 2024. *https://www.healthline.com/health/creatinine-blood#purpose*.

Vapnik, V. and A. Lerner. 1963. Pattern recognition using generalized portrait method. *Automation and Remote Control* 24, 774-780.

Wang, Hongzhi, Mohamed Jaward Bah, and Mohamed Hammad. 2019. Progress in outlier detection techniques: a survey. *IEEE Access* 7, 107964-8000.

World Health Organization. 2021. Cardiovascular diseases (CVDs). Retrieved November 2024. *https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)*.

World Health Organization. 2024. Cardiovascular diseases. Retrieved October 2024. *https://www.who.int/health-topics/cardiovascular-diseases#tab=tab_1*.

Wu, Qiuyi and Ernest Fokoue. 2017. Epileptic seizure recognition [Online]. Available: *https://www.kaggle.com/datasets/harunshimanto/epileptic-seizure-recognition*, doi: 10.13140/RG.2.2.33336.03843.