

Impact of data quality on ML models: Improving data quality with Outlier Detection

by

Rakshit Sharma

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2024

© Rakshit Sharma 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In the dynamic landscape of [Machine Learning \(ML\)](#) applications, data quality comes out to be an important factor that impacts the performance of [ML](#) models. Through this thesis, we present a study that proposes innovative methods for enhancing data quality through an iterative data recapture approach. This research primarily focuses on univariate time-series data where specific patterns can be extracted.

We start by discussing existing data capture methods, where the data is collected manually or using some hardware devices. The proposed methods, namely [Sessionized Recapture Strategy \(SRS\)](#) and [Robust Single Capture Method \(RSCM\)](#), are meticulously detailed, offering distinct strategies for iterative data recapture.

The [Single Capture Method \(SCM\)](#) and [Recapture and Visualize Method \(RVM\)](#) serve as the two baseline methods, with their data capture time and a consequential [False Positive Rate \(FPR\)](#). [SRS](#) is the enhancement of [RVM](#), and [RSCM](#) is the enhancement of [SCM](#). This thesis also introduces an outlier detection algorithm named [Outlier detection through Parameterless Robust Algorithm \(OPERA\)](#), which, when added with [SCM](#) and [RVM](#), results in [SRS](#) and [RSCM](#) respectively.

Compared with the baseline methods, the proposed methods show promising results and improvement in the data quality of the captured data. The experiments are performed on two datasets: one dataset is captured in the Embedded Systems Lab on one of the [ANVIL](#) products for [Future Technology Devices International \(FTDI\)](#) chips, and the second dataset is [Electrocardiogram \(ECG\)](#), provided by PhysioNet and is publicly available [22]. The research concludes with synthesizing key findings and recommendations for practitioners seeking to optimize model performance through enhanced data quality.

Acknowledgements

I would like to thank my supervisor, Professor Sebastian Fischmeister, for his unwavering support and invaluable guidance throughout my journey as a graduate student. I would like to thank you for the motivation and the encouragement that was instrumental in shaping my academic journey at the University of Waterloo. I am deeply grateful to Dr. Carlos Moreno for always being there to help me with my queries and guiding me toward the correct path. Special thanks to Dr. Oleg Iegorov, without whom this thesis would not have been possible. His feedback on my work is invaluable. His expertise guided me through various challenges, ensuring the quality of my research.

Thanks to Srijan Pabbi for having brainstorming sessions with me, Kavish Mehta, Taylor Lee, Sebastien Palmerio, and everyone else from the ESG lab for helping with data collection and lab setups and making work feel like fun.

Dedication

I would like to dedicate this thesis to my mother, Mrs. Ranjana Sharma; my brother, Mr. Prem Sharma; my sisters, Ms. Sonia Sharma and Mrs. Niharika Tejpal Sharma; and my nieces, Ms. Parnika Sharma and Ms. Adviti Sharma.

Table of Contents

Author’s Declaration	ii
Abstract	iii
Acknowledgements	iv
Dedication	v
List of Figures	ix
List of Tables	xi
List of Abbreviations	xii
1 Introduction	1
1.1 Research Motivation	1
1.2 Thesis Outline	2
2 Background	4
2.1 Good Data vs. Bad Data	4
2.2 Challenges in Data Capture	4
2.2.1 Data Capture for FTDI Chips	5
2.2.2 Data Capture for ECG	8

2.3	Fundamental Concepts and Definitions	9
2.4	Existing Outlier Detection Algorithms	11
2.5	Existing Data Capture Processes	13
2.5.1	Single Capture Method (SCM)	13
2.5.2	Recapture and Visualize Method (RVM)	14
3	Methodology	15
3.1	Overview	15
3.2	Exploring Datasets	15
3.2.1	FTDI FT232RL	16
3.2.2	ECG	21
3.3	Outlier detection through Parameterless Robust Algorithm (OPERA) . . .	22
3.3.1	Steps involved in OPERA	22
3.3.2	User Guide	26
	Method description	26
3.4	Proposed Data Capture Techniques	27
3.4.1	Sessionized Recapture Strategy (SRS)	27
3.4.2	Robust Simple Capture Method (RSCM)	29
3.4.3	Convergence Criterion	30
4	Observations	32
4.1	Performance/Data Quality Metrics	32
4.2	Case Study: FTDI	34
4.2.1	SRS based Dataset	34
4.2.2	RSCM based Dataset	37
4.3	Case Study: ECG	39
4.3.1	SRS based Dataset	39
4.3.2	RSCM based Dataset	40

4.4	Comparison between strategies	42
4.4.1	Case Study: FTDI	44
4.4.2	Case Study: ECG	45
5	Conclusion	49
5.0.1	Future Work	50
	References	51
	Glossary	56

List of Figures

2.1	ANVIL by Palitronica Inc. [40]	6
2.2	FTDI FT232RL Chip [18]	7
2.3	Components of a Resting ECG Machine [39]	8
2.4	Histogram vs. Smooth Probability Density Function (PDF) [4]	10
2.5	Effect of various Bandwidth (BW) Values [6]	10
2.6	Empirical Rule [16]	12
3.1	A single S-Parameter for port combination 1,7	17
3.2	A single S-Parameter for port combination 16,10	18
3.3	A single S-Parameter for port combination 1,7 after recapture	19
3.4	A single S-Parameter for port combination 16,10 after recapture	20
3.5	Normal and Abnormal ECG Data	21
3.6	ECG Data with contamination	22
3.7	KDE plot with support range: min(distances),max(distances)	24
3.8	KDE plot with support range: min(distances)-100,max(distances)+100	24
3.9	KDE plot with peaks in red dots	25
4.1	Repeated Data Capture for step 1 in SRS (Blue) and opera.fit output (Yellow)	34
4.2	Data Captured by the user	35
4.3	Miscaptures identified by OPERA	36
4.4	Data after recaptures	36

4.5	Data captured by the user with the output of OPERA.fit method (yellow)	37
4.6	Miscaptures identified by OPERA	38
4.7	Data after recaptures	38
4.8	Randomly selected normal data and its median (representative)	40
4.9	Data captured by the user and output of OPERA.fit method (yellow) . . .	41
4.10	miscaptures identified by OPERA	41
4.11	Data after recaptures	42
4.12	Change in F1-Score with increase in the values of μ for the 3 methods . . .	45
4.13	Change in F1-Score with increase in the values of μ for the 3 methods . . .	47

List of Tables

2.1	Comparison of Good Data and Bad Data [11]	5
4.1	False Positive Rate values using different data capture strategies	39
4.2	FPR values using different data capture strategies	42
4.3	Performance Metrics for Miscapture Rate using SCM	44
4.4	Time to Model (in seconds) vs. μ values	46
4.5	Performance Metrics for Miscapture Rate using SCM	47
4.6	Time to Model (in seconds) vs. μ values	48

List of Abbreviations

AE Auto Encoders [13](#)

BW Bandwidth [ix](#), [10](#), [23](#), [26](#)

CD Correlation Distance [23](#)

chfdb Congestive Heart Failure Database [21](#)

CV Coefficient of Variation [6](#), [7](#)

ECG Electrocardiogram [iii](#), [vi–ix](#), [2](#), [8](#), [14](#), [15](#), [21](#), [27](#), [39](#), [45](#), [48](#), [49](#)

EMI Electromagnetic Interference [9](#)

FN False Negative [33](#), [46](#)

FNR False Negative Rate [33](#), [43](#), [44](#), [46](#)

FP False Positive [33](#), [46](#)

FPR False Positive Rate [iii](#), [xi](#), [33](#), [35](#), [39](#), [40](#), [42–44](#), [46](#)

FTDI Future Technology Devices International [iii](#), [vi–ix](#), [2](#), [5–7](#), [15](#), [16](#), [34](#), [44](#), [45](#), [49](#)

IQR Interquartile Range [11](#), [30](#), [31](#)

ISJ Improved Sheth-Jones [23](#), [26](#)

KDE Kernel Density Estimation [9](#), [10](#), [23](#), [25](#), [26](#), [33](#), [40](#)

MCD Minimum Covariance Determinant [12](#)

ML Machine Learning [iii](#), [1](#), [2](#), [4](#), [13](#), [22](#), [32](#), [42](#), [45](#), [49](#)

OPERA Outlier detection through ParameterlEss Robust Algorithm [iii](#), [vii](#), [ix](#), [x](#), [2](#), [3](#), [22](#), [23](#), [25–30](#), [33–43](#), [46](#), [49](#), [50](#)

OSVM One-Class SVM [12](#)

PDF Probability Density Function [ix](#), [9](#), [10](#)

RSCM Robust Single Capture Method [iii](#), [vii](#), [2](#), [15](#), [29](#), [30](#), [37](#), [39](#), [40](#), [45](#), [46](#), [48](#), [49](#)

RVM Recapture and Visualize Method [iii](#), [14](#), [15](#), [28](#)

SCM Single Capture Method [iii](#), [xi](#), [13](#), [15](#), [29](#), [37](#), [40](#), [44–48](#)

SRS Sessionized Recapture Strategy [iii](#), [vii](#), [ix](#), [2](#), [15](#), [28–30](#), [34](#), [35](#), [37](#), [39](#), [40](#), [45](#), [46](#), [48](#), [49](#)

TP True Positive [33](#)

UART Universal Asynchronous Receiver-Transmitter [16](#)

Chapter 1

Introduction

Today, everybody is trying to extract something useful from the immense data that is currently available. These valuable insights are then used to train various [ML](#) models. Over the last decade, the application of [ML](#) has reached almost all sectors of society, be it banking, medical, engineering, or safety-critical systems.

However, this immense data introduces the need to monitor its quality. Think of a situation where data used to train the [ML](#) models is corrupted. In those cases, the [ML](#) models would learn things that were not intended to.

This raises the importance of ensuring the systems are trained on high-quality data. When the data is collected using hardware systems/devices, there can be a case where the machine malfunctions, environmental conditions impact the data collection, or there is a human impact on the data collection process. All these factors adversely affect the quality of the data capture.

1.1 Research Motivation

This research is driven by the critical importance of data quality in modern [ML](#) applications. The data quality directly impacts the performance of [ML](#) models. Issues such as [miscaptures](#), outliers, and inconsistencies can significantly affect the performance of these models, leading to inaccurate and unreliable results.

One of the main problems resulting in this low-quality data is the data capture process. With this increasing need for data, there needs to be an effective method in place that

can monitor this data quality. In the context of data capture using hardware devices, the traditional method of capturing data just once (thinking that the data was correctly captured) does not identify any of the [miscaptures](#). Another method, where the user uses visual aids to look at the captured data and recaptures it until the user thinks it's correct, takes a lot of effort and time and still isn't reliable. Therefore, we need some innovative techniques and methods that can be adopted to help ensure high-quality data.

Through this research, we try to address these issues by introducing two data capture strategies powered by an outlier detection algorithm called [OPERA](#). We call the first method [SRS](#), which is built upon the method that uses visual aid (discussed above), and then we have [RSCM](#), which is an improvement on the traditional single capture method. Ultimately, our goal is to ensure that the [ML](#) teams get the correct data to do the analysis on, and we aim to achieve this with the suggested approaches.

1.2 Thesis Outline

We start this thesis with an introduction to the underlying problem, which is followed by background information that will help the reader understand this thesis. The background chapter also contains a section that discusses the issues in existing data capture techniques. It mentions key issues in hardware-dependent data capture methods that can result in low data quality. This is followed by a discussion of existing outlier detection methods available in the market. We end this chapter by talking about the current data capture strategies used in the industry.

Chapter 3, titled Methodology, starts with a deep analysis of the datasets pertaining to the two case studies: [FTDI](#) and [ECG](#). We discuss the data quality issue in those datasets by visual inspection. The next section starts with a detailed discussion on [OPERA](#). The step-by-step procedure, its functions, and its pseudo code are mentioned in this section. We also highlight all the critical choices made in this algorithm and provide an explanation. Finally, we discuss the two data capture techniques: [SRS](#) and [RSCM](#). We highlight their strengths, weaknesses, and assumptions made for these techniques.

Chapter 4 starts with a brief introduction to the performance metrics used in the case studies discussed in the next sections. The proposed methods are applied to the two datasets, and the miscaptures identified by these methods are then meticulously examined. The plots shown in this section act as a verification of the output of the proposed methods. Lastly, we compare these techniques by contaminating the train datasets with [miscaptures](#) and analyzing the F1-score with respect to these contaminations.

Finally, in Chapter 5, we conclude the thesis by highlighting the important outcomes of our results on the discussed case studies. We also suggest related work that can be done to further improve [OPERA](#).

Chapter 2

Background

2.1 Good Data vs. Bad Data

When dealing with data capture using hardware devices or systems, keeping all the critical components in perfect working conditions is crucial. These ideal conditions may be their connections, voltage, human intervention, etc. If any vital components are not in place or configured, it might lead to low-quality data or **Bad Data**.

To ensure quality predictions of a [ML](#) model, it depends on whether the data is good or bad [\[11\]](#). Having good-quality data helps organizations make informed decisions. In [Table 2.1](#), let's compare good and bad data.

2.2 Challenges in Data Capture

- **Human errors:** One of the most common challenges we usually encounter in data capture techniques is human error. Whether it is manual data entry, canning, or verification, humans can make mistakes, like typos, omissions, or duplication. This usually results from a lack of interest, fatigue, or distraction. [\[32\]](#)
- **Biased Data:** The data might be selected from larger datasets that don't appropriately convey the message of the broader dataset. Data might be derived from older information that might have resulted from human bias, or perhaps there are some issues with how data is collected or generated that result in a final biased outcome [\[48, 43\]](#). One example of this biased data application is the Amazon Recruitment

Good Data	Bad Data
• Gives correct and reliable information • Collected using calibrated equipment or reliable sources • Well-formatted and structured for analysis. • Consistent and conforms to quality standards • Good data is near to correctness and preciseness • Enhance decision making capability and help taking better decisions	• Might contain incorrect, or inconsistent data • Collected from unreliable sources or using faulty equipment • Poorly formatted, making it difficult to interpret or analyze. • Inconsistent or does not meet quality standards • Bad data lacks precision and accuracy • May result in wrong decisions

Table 2.1: Comparison of Good Data and Bad Data [11]

AI tool [20]. This tool was supposed to select best-match resumes from a pool of applications. This tool was trained on a 10-year-long dataset of resumes, most of which were male candidates' resumes. When this tool was launched, it started rejecting resumes by female candidates. *Note: Here, we are just discussing the issue in the data that impacts the decision-making. Through our work, we are not solving the issue of biased datasets.*

- **Malfunctioned devices:** Another factor that might become a problem in the data capture process is the devices used for data capturing. Any issue with those devices may lead to spurious data entries without even the user being aware of it. This issue might arise due to incorrect cable connections, sending more voltage to the system, or some low-quality parts used in the device.

2.2.1 Data Capture for FTDI Chips

For FTDI chips, we send signals at different frequencies and record the change in the signal as they enter one port of the chip and leave the other. A detailed explanation of how these data/signals look is given in section 3.2.1.

Device for data capture: SilGeo by Moreno et al. [37] is one of the first physics-based hardware assurance tools. Its usage was demonstrated by Vibien in the thesis titled

[SilGeo](#) A Method for the Detection of Counterfeit, Compromised or Tampered Electronic Devices [50].

Figure 2.1 shows [ANVIL](#), a production version of [SilGeo](#), produced by Palitronica Inc. We have used [ANVIL](#) in our thesis to collect data for [FTDI](#) chips.



Figure 2.1: [ANVIL](#) by Palitronica Inc. [40]

Repeatability for [ANVIL](#) setup: To calculate the repeatability of the data capture machine, we performed the following steps:

1. We deliberately perform miscaptures for one target repeatedly. By repeatedly, we mean putting the chip on the setup, letting the data capture complete, removing the chip, putting it again, and so on. We try to introduce an issue in the data capture process for every capture.
2. Next, we calculate the [Coefficient of Variation \(CV\)](#) for the captures performed in the previous step. The [CV](#) is commonly used to assess the precision of a technique [27]. We use this calculated [CV](#) as a baseline for comparison.

$$CV = \left(\frac{\text{standard deviation}}{\text{mean}} \right) \times 100\%$$

3. Now, we randomly pick a target and start repeated captures with no change in the operator, environment or any other factor. In this step, the operator is trained and tries to follow the steps to capture a target.
4. We calculate the [CV](#) for the captures of the new target and compare it with the [CV](#) calculated in step 2.
5. A lower [CV](#) compared to the baseline [CV](#) states high repeatability of the [ANVIL](#) setup.

Steps involved in data capture:

- The [FTDI](#) FT232RL chip is placed on [ANVIL](#)'s adapter board.
- User clicks the 'Start Capture' button, which takes approximately 1 minute to complete the data capture.
- The user then carefully removes the chip from the adapter board and repeats the steps for all the chips.



Figure 2.2: [FTDI](#) FT232RL Chip [\[18\]](#)

Potential known issues in data capture process:

- **Chip placement:** The [FTDI](#) chip might not be correctly placed on the adapter board.
- **Clamp Connection:** User might miss to close the clamp on the chip for a firm hold.
- **Chip Handling:** The [FTDI](#) chips might not have been handled correctly, leading to broken or tilted chips.
- **Adapter Connection:** The adapter board on which the chip is placed might not be connected properly.

2.2.2 Data Capture for ECG

An ECG records the electrical signals in the heart. It's a standard and painless test used to quickly detect heart problems and monitor the heart's health [35].

Device for data capture: ECG for patients is captured using Resting ECG Machines, which are the most commonly used devices in clinical settings. These machines record the heart's electrical activity while the patient rests, providing a baseline assessment of heart health. Resting ECGs typically utilize a 12-lead configuration, capturing the heart's electrical signals from different angles [38]. Figure 2.3 shows the tools/devices used to set up a Resting ECG device in clinics.



Figure 2.3: Components of a Resting ECG Machine [39]

Repeatability for Resting ECG Machine setup: The reproducibility and repeatability of resting ECG machines are discussed by Huang et al. [24] and by Gellert et al. [19]. Both papers conclude that these machines have high reproducibility and repeatability.

Steps involved in data capture:

- The Patient is first asked to remove any jewelry or metal from their body that might interfere with the readings.

- Attach electrodes to the patient’s body per standard ECG lead placement.
- Skin should be clean and dry before attaching electrodes.
- Ensure the connections are secure and then start the recording.

Potential known issues in data capture process:

- **Incorrect placement of electrodes:** Wrong placement of electrodes leads to incorrect interpretation and misdiagnosis of conditions [36, 30, 31, 7].
- **Electromagnetic Interference (EMI):** EMI can lead to incorrect captures of ECG signals. This EMI is usually generated by mobile phones when placed too close (7.5 cm.) to the electrodes [10, 41].

2.3 Fundamental Concepts and Definitions

- **Outlier and Anomaly:** An Anomaly is an unusual, unexpected, or surprising pattern in the given data. Outliers, discordant observations, exceptions, aberrations, or contaminants are different synonyms of the term ‘anomaly’, used in different fields and applications [12].
- **PDF:** In statistics, a PDF is a mathematical function that delineates the association between a random variable and its probability. This function allows for the determination of the probability of the variable [46].
- **Kernel Function:** A kernel function estimates the density of random variables. A kernel function should satisfy the following properties:
 - It must be symmetrical.
 - The area under the curve of the function must be equal to one.
 - The value of the kernel function (density) cannot be negative.
- **Kernel Density Estimation (KDE):** KDE is the process of estimating an unknown PDF using a kernel function. A kernel density estimate is a function defined as the sum of a kernel function on every data point [1].

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

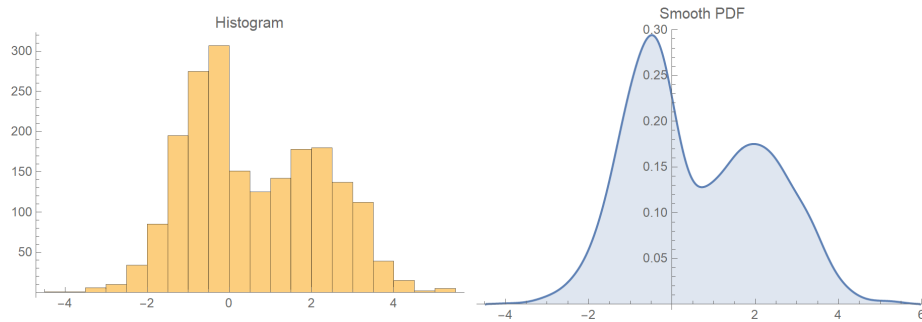


Figure 2.4: Histogram vs. Smooth PDF [4]

- **BW Parameter in KDE:** The BW parameter is used to define how wide the density lumps are. [3] It is similar to the histogram bin width. As shown in Figure 2.5, the smaller the BW value, the noisier the data would look than it actually is. Similarly, larger values of BW can result in over-smoothing of the curves. Choosing the appropriate value of BW is crucial to visualize the data.

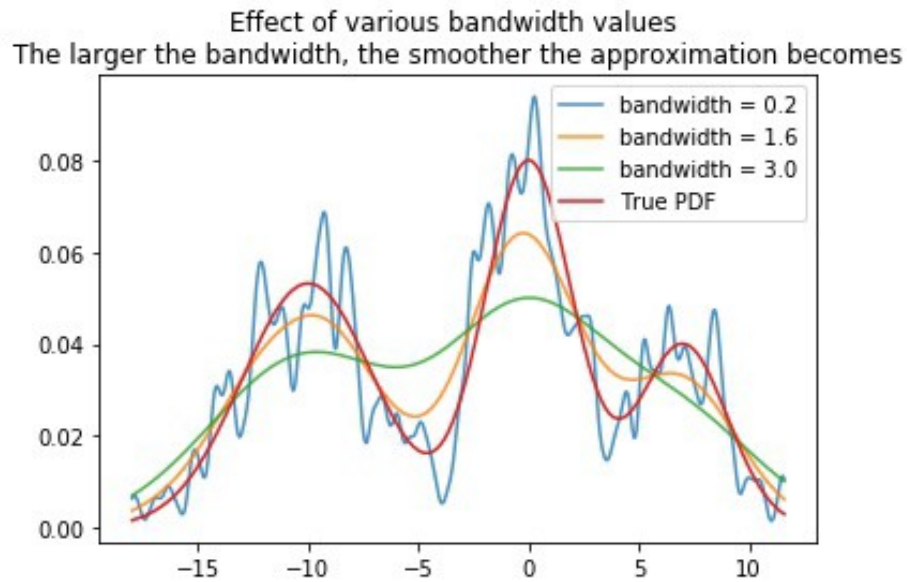


Figure 2.5: Effect of various BW Values [6]

- **Time Series Analysis:** Time Series Analysis is a special way to analyze data collected over time. Usually, this analysis is performed to capture different patterns

in the data.

- **Anomalies in Time Series:** When data deviates from the observed pattern in time series data, that data point is marked as an anomaly. In simple words, think of a line graph, with the x-axis as time and the y-axis as some feature/property. Most of the data would follow a specific pattern, which we can say is the fingerprint of the dataset. If any data point deviates from this fingerprint, then we flag that data point as an anomaly.

2.4 Existing Outlier Detection Algorithms

1. **Interquartile Range (IQR) Method:** Box Plots are usually used to visualize how the data is distributed. It plots Q1 (25th percentile), Q2 (50th percentile or median), Q3 (75th percentile), lower and upper bounds, which are generated using IQR, which is the difference between Q3 and Q1.

- Lower Bound: $LB = Q1 - 1.5 \times IQR$
- Upper Bound: $UB = Q3 + 1.5 \times IQR$

Any data point above or below these upper or lower bounds is marked as an outlier [23].

2. **Z-Score Method:** This method is used for outlier detection when the variable's distribution looks like Gaussian. The Z-score indicates how many standard deviations a particular value deviates from the mean of its variable [5].

$$Z = \frac{(X - \mu)}{\sigma}$$

where X is the value being measured, μ is the mean of the population, and σ is the standard deviation of the population.

To use a z-score for outlier detection, the user defines a cut-off or threshold value, outside of which any value is marked as an outlier. Usually, this value is chosen as -3 and +3, as 99.7% of the values lie within this range in the standard normal distribution as per Empirical Rule [16] (see fig 2.6).

3. **Isolation Forest [33]:** Isolation Forest is a tree-based anomaly detection algorithm. This algorithm tries to isolate anomalies that are few in number and different from

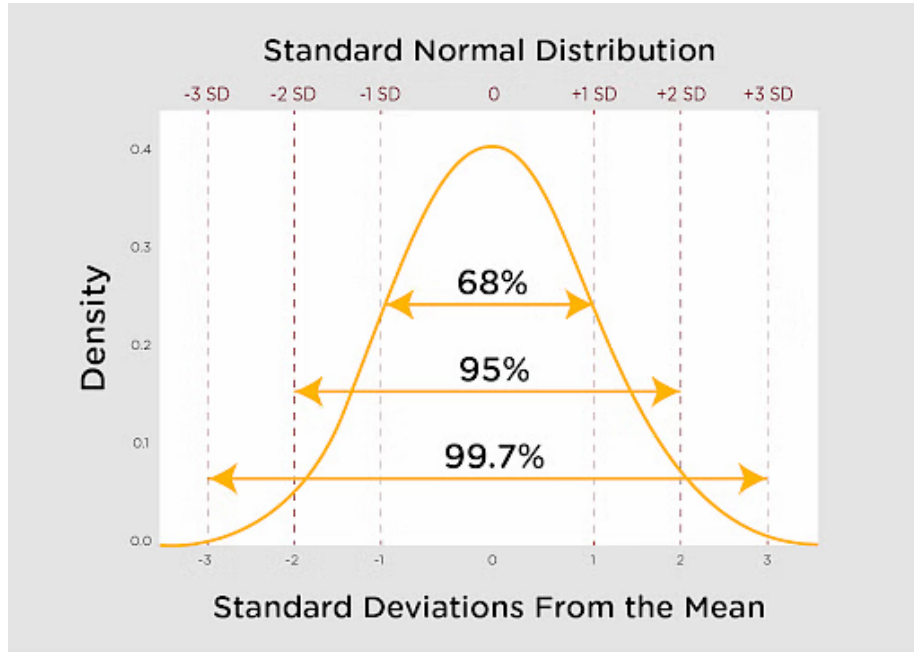


Figure 2.6: Empirical Rule [16]

the normal data. It requires an important hyper-parameter called *contamination*. Its value ranges between 0.0 and 0.5. It helps the algorithm set the threshold for anomaly detection. It is the percentage of anomalies that can be expected in the training dataset.

The only drawback here is that, in many cases, we only have normal data for training, but this approach requires both normal and anomalous data, even for training.

4. **Minimum Covariance Determinant (MCD)** [25]: This approach works if the input variables have Gaussian distribution. In **MCD**, it tries to create a hyperspace or ellipsoid that surrounds the normal data, and data outside this shape are then flagged as outliers. This approach also requires *contamination* value, corresponding to the percentage of outliers inside training data.
5. **One-Class SVM (OSVM)** [42] **OSVM** is based on the principle of support vectors that act as a boundary to differentiate between classes. In the case of **OSVM**, the algorithm captures the density of the majority class, and any data point outside this boundary is identified as an outlier. **OSVM** assumes that most or all of the train dataset data is normal. This algorithm requires a *kernel* hyper-parameter

(e.g., linear, polynomial, Radial Basis Function), and each *kernel* has its own hyper-parameters.

6. **Auto Encoders (AE)** [14, 17, 49] AE are used to reconstruct the input that it receives. An AE has two components: encoder and decoder layers. The encoder layer tries to reduce the dimensions of the input vector by trying to learn and generalize the patterns present in the data. The decoder layer then tries to regenerate the input vector using the reduced dimensions.

This definition can be extended to the application of anomaly detection. The AE is trained on normal data and is expected to understand the patterns and details of this normal data and should, ideally, regenerate this input normal data with some reconstruction error. When anomalous data is sent to this trained autoencoder, it tries to regenerate the input but with a higher reconstruction error. Hence, a reconstruction error threshold needs to be defined, above which all the data points would be marked as outliers. The limitation of this approach is that we require a lot of data for these AE, which is sometimes impossible in real-world applications.

2.5 Existing Data Capture Processes

2.5.1 Single Capture Method (SCM)

The SCM involves a single capture for a target. During this capture session, the target and the hardware device are attached together, data gets captured, and the target is detached from the data capture device. This process continues until the user is done capturing all the targets. This is one of the current methods used to capture the data in the industry [8].

Strengths

- It takes less time for data capture than other methods.
- This method is easier for the user as they are required to capture the target only once.

Weaknesses

- The user is unaware if the recorded data for a target is correct.
- Data capturing using SCM may lead to low-quality ML models if the dataset for model training has any issues.

2.5.2 Recapture and Visualize Method (RVM)

The [RVM](#) involves data capture and visualization of the captured data for the [targets](#) in the data capture process. In this process, the user usually has some data visualization tools, depending on the nature of the data. While capturing this data, the user can see how it looks, and if it's not correctly captured (as per user), then they can recapture that target at that point in time.

Consider a case when a nurse or an intern is taking [ECG](#) readings of a patient. There can be cases of wires not being correctly plugged in, electrodes not being in place, or the reasons mentioned in section 2.2.2. When the data is captured by not-so-trained users, this visualization helps them to be confident that they are doing things the right way. In [ECG](#) capture, the supervisor also needs to start recording the readings that they think are useful by looking at the event monitor [\[28\]](#).

Strengths

- It gives confidence to the user regarding the data quality.
- It can help users guide and follow a specific process for that data capture.

Weaknesses

- It can take a very long time because, for every capture, the user needs to take a look at the data.
- There is a user interference where the user decides whether the captured data is correct or not based on the visualization. It introduces the factor of human error.

Chapter 3

Methodology

3.1 Overview

In this section, we will deep dive into the proposed methods to increase data quality, comparing them with the existing methods discussed in section 2.5 and listing their merits and demerits. We propose two methods, namely [SRS](#) and [RSCM](#). The [SRS](#) method specifically comprises 2 main steps: **Preparing a good representation of data** and **Data Quality Assurance**. [SRS](#) is based on the ideology of [RVM](#) of multi-capture of data, whereas [RSCM](#) is an enhancement to the [SCM](#). In [SRS](#), we try to automate the step where the user takes the help of visualization tools to analyze the captured data in [RVM](#). Specifically, the first step of [SRS](#) helps to automate this by asking the user to perform multiple captures for some random initial [targets](#).

3.2 Exploring Datasets

This section highlights different datasets that were used in experimentation. We used [FTDI](#) FT232RL chips for our first case study. This choice of using [FTDI](#) FT232RL chips is based on the fact that these chips are easily available in the market and are cheaper in price. The data for [FTDI](#) was collected using the [ANVIL](#), as discussed in section 2.2.1. For the second case study, we used [ECG](#) data, which is publically available in the PhysioNet database [22], as discussed in section 3.2.2. All nominal data for each test case is expected to have a similar structure or fingerprint(allowing some variability). If any of the data points look different from the rest of the nominal data, then there can be two cases: either

the data is an outlier, or the data was not captured correctly. It's important to note that we are not trying to find the issue of the [miscaptures](#). The main motive is to flag the [miscaptures](#) and ask the user to recapture that target.

3.2.1 FTDI FT232RL

The [FTDI FT232RL](#) is a popular USB-to-serial [Universal Asynchronous Receiver-Transmitter \(UART\)](#) interface converter chip. We collected the data for 144 [FTDI](#) chips, out of which 74 are labeled as authentic and 70 as counterfeit chips. These chips were ordered from various vendors, including Mouser, Digikey, Ali-Express, and vendors on eBay. Ordering chips from multiple vendors helped us get more dataset variation.

Upon visual inspection of the data recorded by [ANVIL](#), we saw that out of 74 authentic chips, 13 looked different from other nominal chip data. Figure [3.1](#) and [3.2](#) shows Logarithmic Magnitude and Group Delay for some port combinations for this data. Please note that Logarithmic Magnitude (S11 and S12) and Group Delay (G11 and G12) are this dataset's features, which are discussed in detail by Vibien in this master's thesis [\[50\]](#).

If all the chips in this dataset were nominal, then they should be close to each other, which is not the case, as is clearly visible in Figure [3.1](#) and Figure [3.2](#).

Using visual inspection of the graphs, we listed 13 chips' data that looked different from the whole set of nominal chips. As discussed in the previous section, this can mean it is either an outlier or a miscapture. To verify this, we collected the data for these 13 chips again.

Figures [3.3](#) and [3.4](#) show the data for the same port combinations, but this time, after recapturing those 13 chips. If we compare the graphs in fig [3.1](#) and fig [3.3](#) and the graphs in fig [3.2](#) and fig [3.4](#), it is evident that the variation in the data is gone. This states that the issue was in the data capture process, thereby eliminating the possibility that those were outliers.

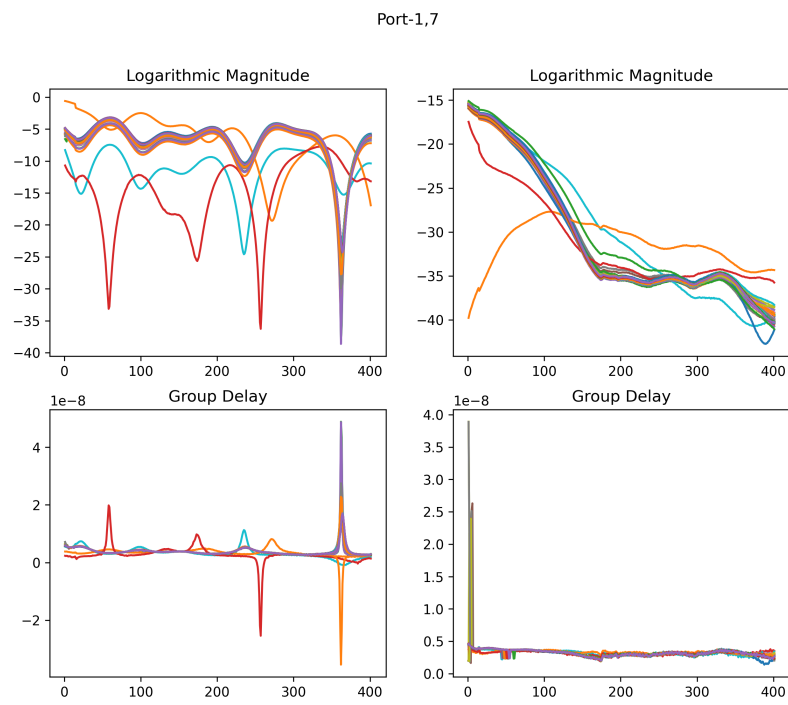


Figure 3.1: A single S-Parameter for port combination 1,7

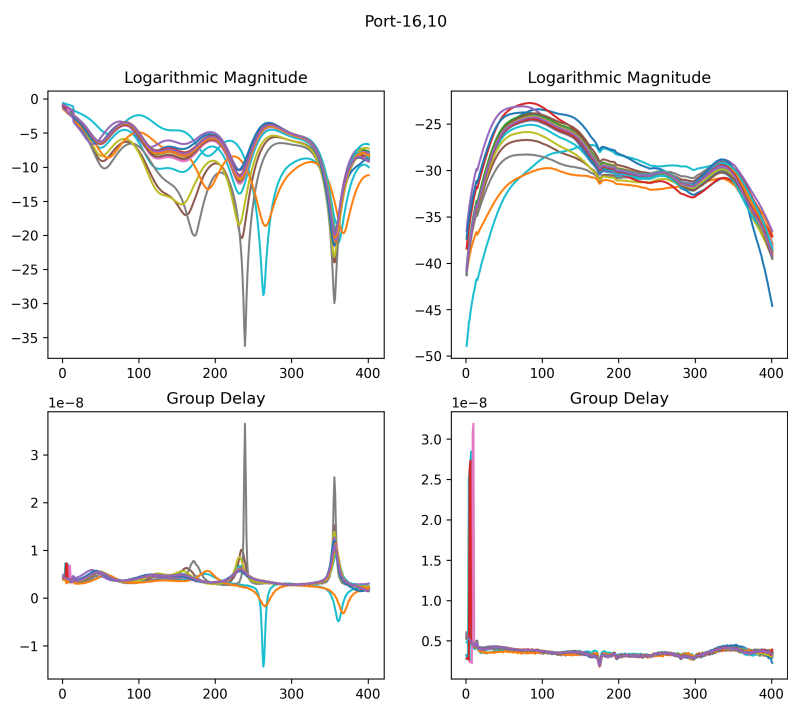


Figure 3.2: A single S-Parameter for port combination 16,10

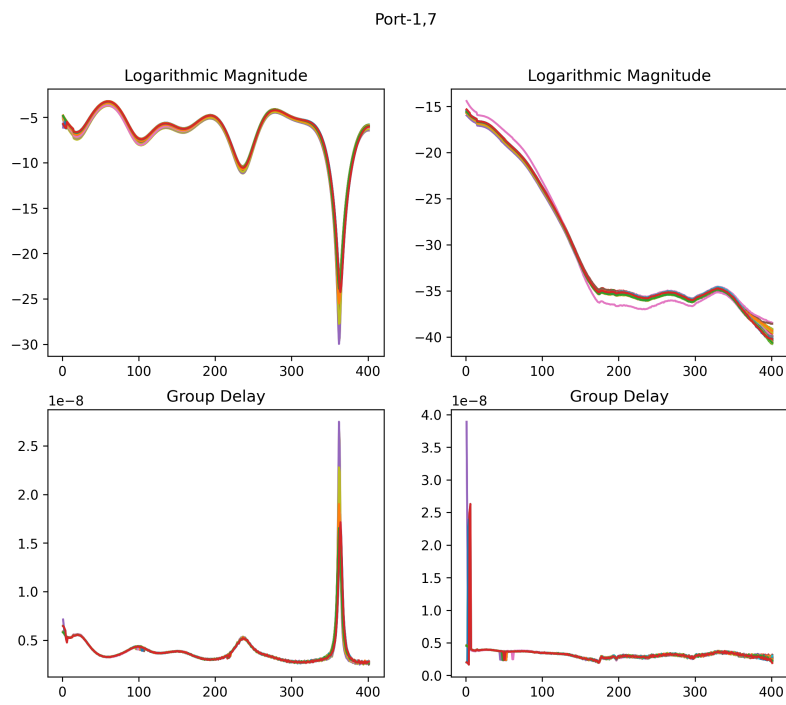


Figure 3.3: A single S-Parameter for port combination 1,7 after recapture

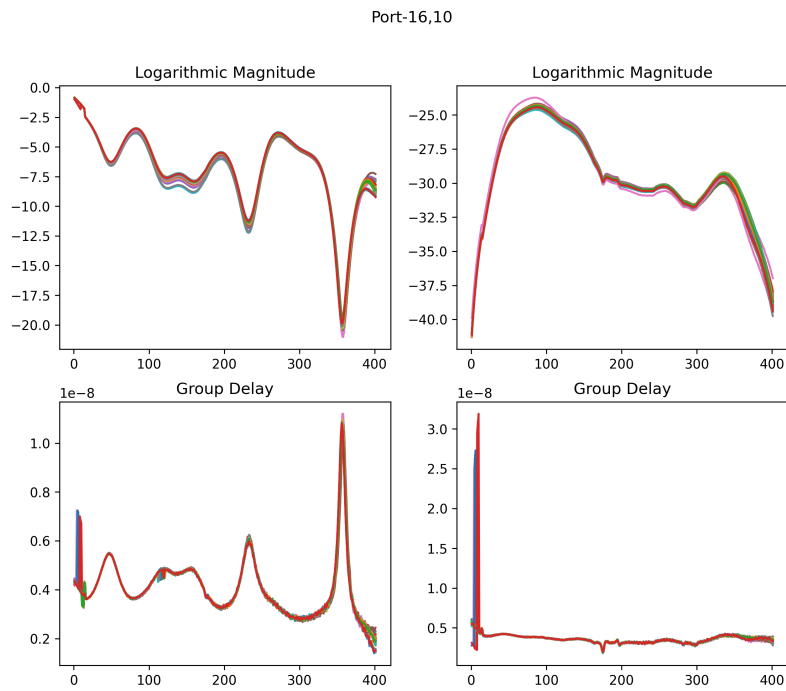


Figure 3.4: A single S-Parameter for port combination 16,10 after recapture

3.2.2 ECG

The ECG dataset [22] originates from a 20-hour-long ECG obtained from the PhysioNet database’s BIDMC Congestive Heart Failure Database (chfdb). This dataset was initially featured in a publication by Goldberger et al. [21] in the Circulation journal.

For the data preparation, a two-step preprocessing approach was employed. Initially, each individual heartbeat was extracted from the raw recording, followed by an interpolation to standardize the length of each heartbeat. This curated dataset, utilized by Chen et al. [13], comprises 5,000 randomly selected heartbeats.

It’s noteworthy that the patient from whom the ECG data originated presented severe congestive heart failure. Automated annotation processes were utilized to assign class labels to each heartbeat. The resulting dataset that we use for our study comprises 2,080 heartbeats labeled as abnormal and 2920 heartbeats labeled as normal.

As discussed in section 2.2.2, in order to simulate the scenario, we will try to contaminate the normal ECG data with the abnormal one, and in Chapter 4, we will dig deeper into the results of our method to see if it can identify those contaminations.

- Figure 3.5 shows the normal and abnormal ECG data, respectively.
- Figure 3.6 shows the ECG data when it is contaminated with randomly selected abnormal ECG data.

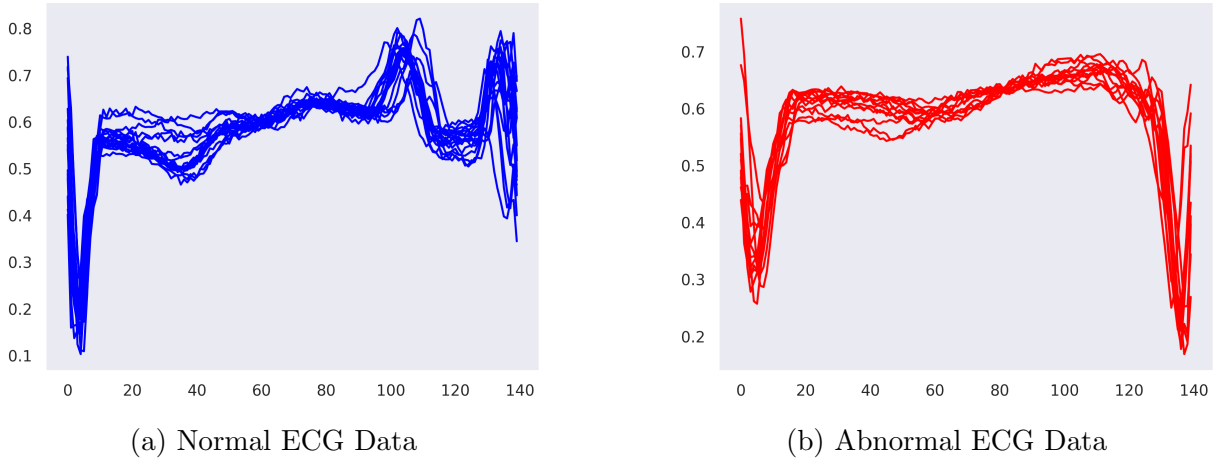
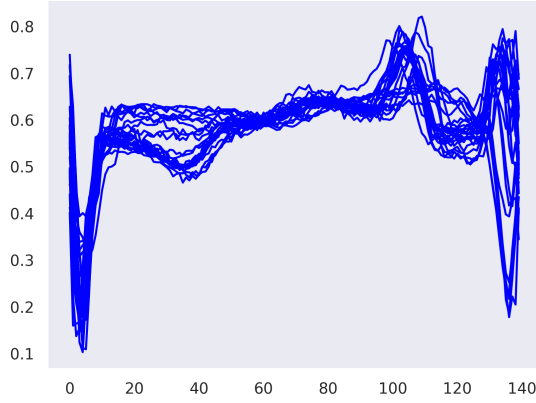
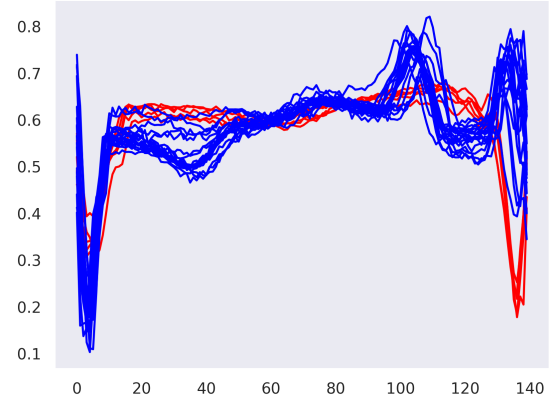


Figure 3.5: Normal and Abnormal ECG Data



(a) ECG data with contamination



(b) ECG data with contamination (marked)

Figure 3.6: ECG Data with contamination

3.3 Outlier detection through Parameterless Robust Algorithm (OPERA)

In this section, we propose an algorithm called **OPERA** for identifying **miscaptures** in given data. **OPERA** is based on the assumption that the data comes from the same underlying distribution.

Parameterless?

The main highlight of **OPERA** is that it is parameterless. This means that unlike other **ML** methods that usually require hyper-parameter tuning for different datasets, **OPERA** does not require any of this tuning.

3.3.1 Steps involved in **OPERA**

1. **Median Representation Calculation:** We calculate one reference of the given dataset. The median of each feature (along the columns of the dataset) is calculated. This results in a vector that captures the median of the data without being influenced by extreme values.

2. **Distance Calculation:** Using the median as a reference, we compute the distance of each data point from the median. We choose **Correlation Distance (CD)** as the distance metric. **CD** measures the dissimilarity between two vectors while accounting for differences in scale and location. The choice of the distance metric is explained at point 5.
3. **Estimating the distribution of distances:** **OPERA** uses **KDE** to estimate the distribution of distances, calculated in step 2. In the formula mentioned in section 2.3, we need to assign a **BW** i.e. ‘h’, and there are various methods to estimate that or manually assign. To make **OPERA** parameterless, we use the **Improved Sheth-Jones (ISJ)** method proposed by Botev et al. in [9]. **ISJ** method does not assume the distribution of $\hat{f}_h(x)$ to be normal, unlike other **BW** estimation methods. **ISJ** is a data-driven approach that adapts the **BW** to the underlying distribution, minimizing the risk of oversmoothing or undersmoothing. Iegorov and Fischmeister use the **ISJ BW** selection method to perform modal-based clustering [26]. In our case, we use this proposed approach in [26], but we are only concerned with the leftmost cluster, i.e., the cluster that is closer to 0 on the x-axis.
4. **Support Range Selection for KDE** The support range selection is an important factor in estimating the distribution using **KDE**. Usually, this range is chosen to lie between the minimum and maximum values present on the x-axis. But if we increase this range, then the resultant shape will contain fewer clusters, as compared to a shorter range (between the minimum and maximum of the values), as shown in fig 3.7 and fig 3.8.

At point 2, we talked about **CD** as the distance measure to be used to calculate the distance between the median and other **targets**.

$$\text{correlation_distance}(X, Y) = 1 - \text{correlation_factor}(X, Y)$$

where

$$\text{correlation_factor}(X, Y) = \frac{\text{cov}(X, Y)}{\sqrt{\text{var}(X) \cdot \text{var}(Y)}}$$

$$\text{cov}(X, Y) = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})$$

$$\text{var}(X) = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2$$

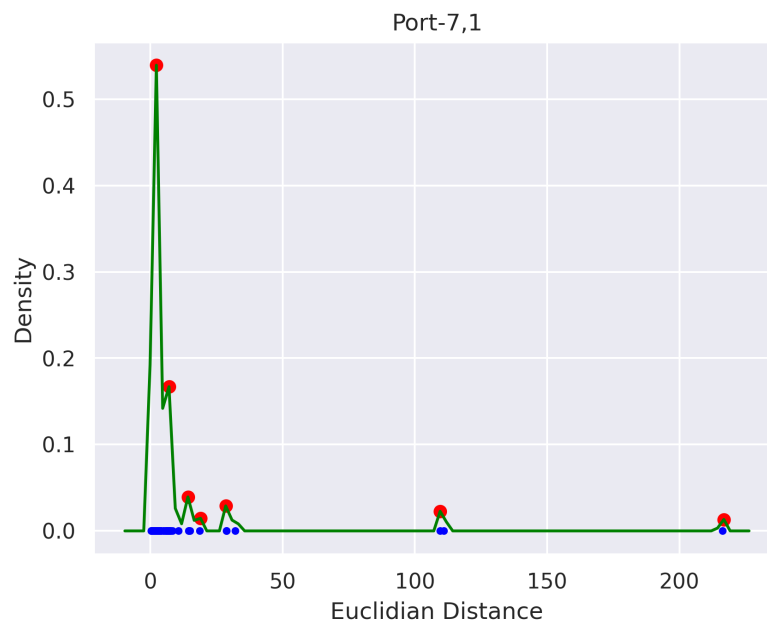


Figure 3.7: KDE plot with support range: $\min(\text{distances}), \max(\text{distances})$

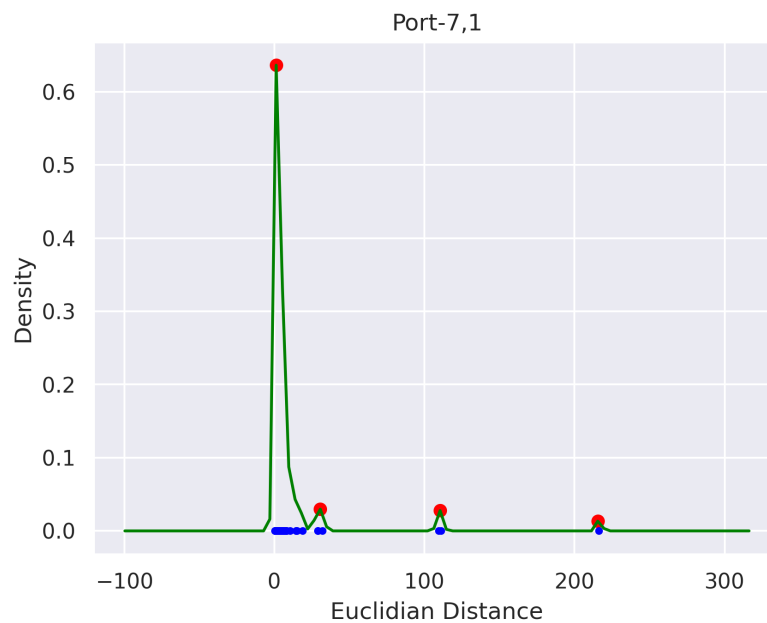


Figure 3.8: KDE plot with support range: $\min(\text{distances})-100, \max(\text{distances})+100$

The correlation_factor lies in the range of $[-1,+1]$, which results in the correlation_distance being in the range of $[+0,+2]$, resulting in a bounded support range for KDE (see fig 3.9). In other distance measures, this value won't be bounded and we would have to set different support range for different datasets.

5. **Cluster Identification and assignments:** Now, we find the clusters in KDE plot by calculating local maximas, as shown in fig 3.7, 3.8 and 3.9 with red dots. OPERA identifies outliers as targets whose distance value does not belong to the leftmost cluster of the KDE plot. The cluster to which a data point on the x-axis belongs to depends upon 2 factors:

- How close the data point is to a cluster. Here, we only consider the distance in 1-dimension (x-axis).
- If there is a local minimum between the data point and its nearest peak, we assign the next nearest peak as the cluster to which this data point belong to.

These outliers represent deviations from the typical data distribution and are flagged for recapture.

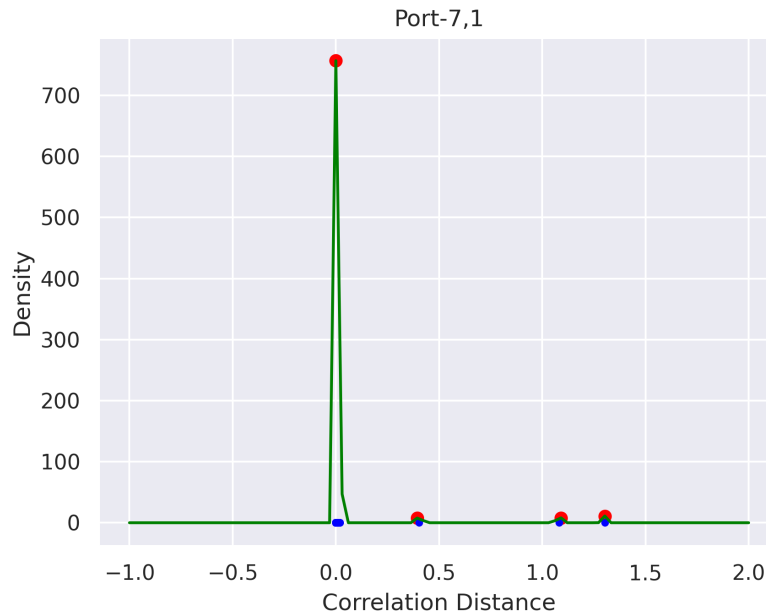


Figure 3.9: KDE plot with peaks in red dots

3.3.2 User Guide

Parameters:	None
Attributes:	median: <i>ndarray of shape (1, n_features)</i>
Methods:	• fit(X: ndarray of shape (number of data points,n_features)): Calculates the median of X and saves in <i>median</i> • predict(X: ndarray of shape (number of data points,n_features)): Calculates the outliers in X.

Method description

- **fit**(X): This method takes input an ndarray of shape (number of data points,n_features). The median of the input is then computed and saved in the *median* attribute of [OPERA](#) (check algo 1).
- **predict**(X): This method takes input an ndarray of dimension (number of data points, n_features). It computes the distance of each data point from the median, and estimates the distribution of these distances using [KDE](#) with [ISJ](#) as [BW](#) method. The clusters in the distribution are then computed, and each distance value (used for estimating the [KDE](#)) is assigned to its nearest cluster. The data points for which their corresponding distance value does not belong to the leftmost cluster are returned by this method (check algo 2).

Algorithm 1: fit

Data: X: ndarray of shape (n_samples, n_features)

1 median $\leftarrow$ X.median()

Algorithm 2: predict

Data: X of shape $(n_samples, n_features)$

Result: ndarray of data points identified as outliers

- 1 $distances \leftarrow \text{calculate_distances}(\text{median}, X)$
 - 2 $X_i, Y_i \leftarrow \text{KDE}(\text{data} = \text{distances}, \text{bandwidth_method} = \text{'isj'})$
 - 3 $Y_{\text{peaks}} \leftarrow \text{calculate_peaks}(X, Y)$
 - 4 $assignments \leftarrow \text{assign_peak_to_distances}(\text{distances}, Y_{\text{peaks}})$
 - 5 $outliers \leftarrow \text{items where assignment} \neq 1 \text{ in assignments}$
 - 6 **return** *outliers*
-

Algorithm 3: calculate_outliers (Example)

Data: $\Gamma_{\text{train}}, \Gamma_{\text{test}}$

Result: outliers detected by OPERA

- 1 $\text{opera} \leftarrow \text{OPERA}();$
 - 2 $\text{opera.fit}(\Gamma_{\text{train}});$
 - 3 $\text{outliers} \leftarrow \text{opera.predict}(\Gamma_{\text{train}} + \Gamma_{\text{test}});$
 - 4 **return** *outliers*
-

3.4 Proposed Data Capture Techniques

3.4.1 Sessionized Recapture Strategy (SRS)

In section 3.1, we touch base on two topics: Preparing a good representation of data and testing the data captured by the user. In this section, we will discuss both of these topics.

1. **Preparing a good representation of data:** A ‘good quality’ dataset means that the data was captured in the most optimal conditions, it has the least human error possible, and the environment or the surroundings are stable. In this part, we introduce a process that should be followed by the user when the user captures the data. Below is the step-by-step procedure:

- For case study 1, ask the user to put the chip on [ANVIL](#) and click the ‘Start Capture’ button on the screen. Similarly, for the latter case, ask the user to take the [ECG](#) reading for the patient.
- Once the capture is complete, ask the user to do this capture 5 times for the same target. Here, we assume that these certified data capture machines would

be able to capture the data correctly for most cases. For our case studies, this assumption is supported with the discussion in section 2.2.1 and section 2.2.2 .

- This process is repeated for 5 randomly selected **targets**. Based on the above-mentioned assumption, we calculate the median of the 5 captures for each of the 5 **targets**. Since most of the captures of a single target would be the correct captures, the calculated median would correctly represent that particular target.

It takes a lot of time to perform multiple captures of a single target and repeating it for multiple targets, so we chose to capture 5 **targets** 5 times in our case. Capturing 5 **targets** 5 times gives us 25 data points, and after calculating the median for each target, we are left with 5 data points that represent the ‘good representation of data’.

2. **Data Quality Assurance:** The **SRS** builds upon **RVM** by introducing an automated iterative approach to data capture. Algorithm 4 provides the pseudocode for the **SRS** algorithm, which helps detect any **miscaptures** in the data capture process.

After performing Step 1, we get the ‘good representation of data’, and that data is used as an input for OPERA.fit method (Γ_{train}). Now, we ask the user to start capturing the remaining **targets** one by one till the user is done capturing data for all the **targets**. Γ_{train} , along with the data for these newly captured **targets** acts as an input for **OPERA**.predict method ($\Gamma_{\text{train}} + \Gamma_{\text{test}}$).

- We call OPERA.fit with Γ_{train} as input to the function.
- OPERA.fit calculates the median for the input data and saves it in the *median* attribute of **OPERA** class.
- Then we call opera.predict with ($\Gamma_{\text{train}} + \Gamma_{\text{test}}$) as the input to the function.
- opera.predict returns the data points identified as outliers.

Algorithm 4: **SRS**

Data: $\Gamma_{\text{train}}, \Gamma_{\text{test}}$

Result: outliers detected by **OPERA**

```

1 opera ← OPERA();
2 opera.fit( $\Gamma_{\text{train}}$ );
3 miscaptures ← opera.predict( $\Gamma_{\text{train}} + \Gamma_{\text{test}}$ );
4 return miscaptures

```

Strengths

- This method ensures that the initial data captured is correct.
- Data capturing using [SRS](#) would lead to high-quality data.

Weaknesses

- User needs to give more time in the data capture process.
- We are forcing the user to change their data capture process.
- [SRS](#) is based on the fact that the data capture device has high repeatability.

3.4.2 Robust Simple Capture Method (RSCM)

In [RSCM](#), we let the user capture all the [targets](#) (similar to what is done in [SCM](#)) and use data for all the [targets](#) as input to the OPERA.fit method.

The motivation behind [RSCM](#) is to ensure that the user experience is not impacted, which was the case in [SRS](#). The process for data capture in [RSCM](#) is same as that [SCM](#) discussed in section 3.4.1, with an addition of [OPERA](#). In this method, once the user captures all the [targets](#), we trigger Algorithm 5 which returns the [miscaptures](#) identified by [OPERA](#). The user is then asked to recapture those [targets](#) that the algorithm returns.

Algorithm 5: [RSCM](#)

Data: Γ

Result: outliers detected by [OPERA](#)

```

1 opera  $\leftarrow$  OPERA();
2 opera.fit( $\Gamma$ );
3 miscaptures  $\leftarrow$  opera.predict( $\Gamma$ );
4 return miscaptures

```

Strengths

- Data capturing using [RSCM](#) would lead to high-quality data.
- This method takes less time than [SRS](#).
- There is no change in the data capture process, as in the case in [SRS](#).

Weaknesses

- [RSCM](#) is based on the fact that the data capture device has high repeatability.

3.4.3 Convergence Criterion

Below are the steps followed after our method flags a target to be recaptured.

1. **Initial Capture:** The system flags a target as a miscapture, and the user is asked to recapture it.
2. **Recapture and KDE Analysis:** The distance of the recapture from the previously computed median (calculated by [OPERA](#)) is calculated, and the distribution for all distance values is estimated using KDE.
3. **Stopping Criteria based on KDE:**
 - **Correct Capture:** If the recapture gets clustered together with the “correct captures” cluster, the process ends, and the capture is considered valid.
 - **True Outlier:** If the recapture gets clustered together with the data point for the previous recapture, we capture it one more time meaning that we try to see if 3 consecutive recaptures fall within the same cluster, we can say it’s a true outlier. This highlights that there was some issue in the data capture, and the user identified the issue and fixed it, but the target is still an outlier.
4. **Uncertain Cases (if stopping criteria not met after 30 recaptures - based on Central Limit Theorem [2]):**
 - **Calculating [IQR](#):** If the recapture distances don’t meet the initial stopping criteria, We calculate the [IQR](#) for the following cases to capture the variation:

- The recapture distance values (including all attempts).
- The distance values of “correct captures” (data points from the leftmost cluster in the KDE plot).

$$\text{IQR} = Q_3 - Q_1 \quad (3.1)$$

*Note: The distribution of the distances of the correct captures is not Gaussian hence, we cannot compute a confidence interval, and **IQR** works for any distribution*

- **Outlier vs. Capture Process Issue:** There can be two types of variations:
 - Variation within targets (IQR_{vt}).
 - Variation within captures of one target (IQR_{vc}).
 - If the IQR_{vc} for the recaptures is less than the IQR_{vt} for the “correct captures”, the target is likely a true outlier.
We can say this because the variation in the “correct captures” distances is due to the variation between the targets, whereas the variation in the recapture distances is due to the variation within the captures of a single target. If the data is being captured correctly, the variation in the latter case would always be less than in the former case (supported by the repeatability of the data capture machine).
 - If the IQR_{vc} is wider, it indicates an issue with the capture process itself, calling it a miscapture.

Chapter 4

Observations

4.1 Performance/Data Quality Metrics

In our study, we are particularly focused on the effect of data quality on ML models and how to mitigate this issue. When we try to add an additional step during the data capture process, the time and effort of the user increases. So, a trade-off between time and ML model quality must exist.

In the upcoming sections, we will look at two case studies about the datasets which we discussed in section 3.2. Here, we will see how the miscaptures percentage in the training data impacts the ML model quality and how our proposed methods help to improve it. At the same time, we will also look at how the time factor comes into the picture once our proposed methods are implemented.

- The Miscapture Percentage (μ) in the training data is calculated using:

$$\mu = \frac{\text{Number of miscaptured items}}{\text{Total number of items}} \times 100$$

- Assuming that the time taken to capture one item on a hardware device is ν seconds and the total number of items to be captured by the user is ∇ , then the total time taken would be:

$$\theta_{\text{total}} = \nu \times \nabla$$

In the case studies discussed below, we would be using F1-Score as the measure to quantify the model's quality when it is trained on datasets with different μ values.

We will be using OPERA as the anomaly detector here by calculating the median of the train data set, which contains only nominal data, and using a test dataset that contains both nominal and anomalous data to plot KDE, calculate & assign peaks, and find anomalies.

In anomaly detection, when we say that a prediction is False Positive (FP), it means that the particular data point in the test dataset was actually nominal but has been predicted as an anomaly by the model. Similarly, if a prediction for a data point is False Negative (FN), it means that the data point is actually an anomaly but the model incorrectly predicts it to be a normal data point.

- **FPR:** FPR is defined as the ratio of FP predictions to the actual negative instances.

$$\text{FPR} = \frac{FP}{FP + TN}$$

- **False Negative Rate (FNR):** FNR is defined as the ratio of FN predictions to the actual positive instances.

$$\text{FNR} = \frac{FN}{TP + FN}$$

- **Precision:** It is the proportion of True Positive (TP) out of all the positive predictions made by the model.

$$\text{precision} = \frac{TP}{TP + FP}$$

- **Recall:** It is the proportion of actual positive cases (TP) that were correctly identified by a model out of all the actual positive cases in the dataset.

$$\text{recall} = \frac{TP}{TP + FN}$$

- **F1-score:** F1-score or F-measure is the harmonic mean of precision and recall. The F1-score gives equal importance to both precision and recall.

$$\text{F1-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

A higher F1-score suggests a better balance between precision and recall. It means that the model is both accurate and able to capture a high proportion of positive events.

4.2 Case Study: FTDI

4.2.1 SRS based Dataset

Here, we deploy our [SRS](#) method by first asking the user to do 5 captures of 5 randomly selected chips and calculate the good representation of these 5 chips (refer [3.4.1](#)). Then, the median of the good representation is calculated, the user is asked to capture the rest of the chips, and we find the distance of the data point for each chip that the user captured from the calculated median. If any outlier is identified by [OPERA](#), the user is asked to capture it again. Even after asking the user to capture again, if the same [targets](#) are being asked to recapture again, we can say it is a definite outlier.

Figure [4.1](#) shows the data points for the [targets](#) that the user captured for step 1. Here, the user captured 5 randomly selected FTDI chips 5 times. The blue lines show the data for those 5 chips, and the yellow line shows the output of the fit method of [OPERA](#).

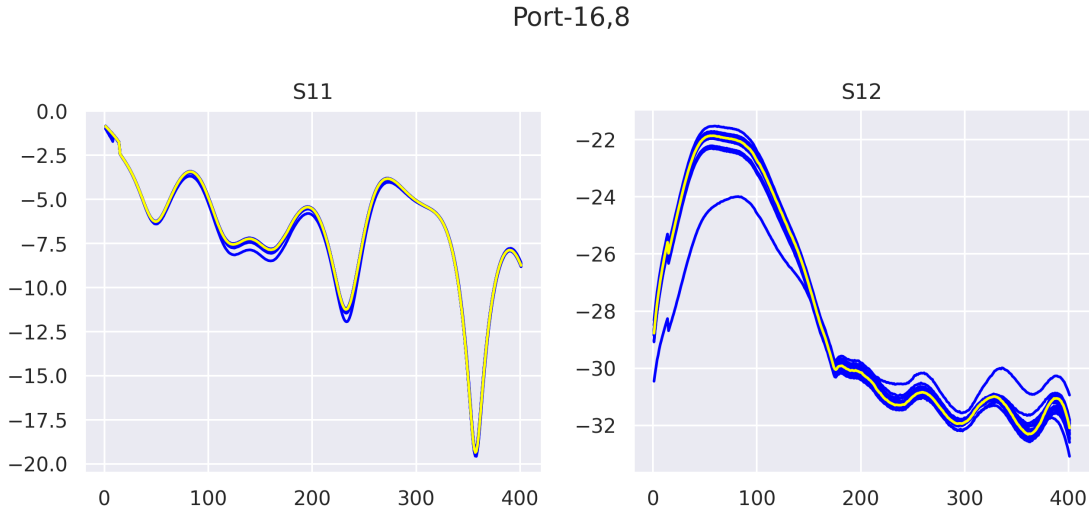


Figure 4.1: Repeated Data Capture for step 1 in [SRS](#) (Blue) and opera.fit output (Yellow)

Figure [4.2](#) shows the data points for all the remaining chips that the user captures, and fig [4.3](#) shows the data points for all those remaining chips with red data points identified as [miscaptures](#) by the algorithm. The user is asked to capture them again, and fig [4.4](#) shows the data points after the user completes the recaptures of the chips that the algorithm returned as [miscaptures](#).

As it is clearly seen, the algorithm is able to identify the data points which are [miscaptures](#) (Fig 4.3), and upon recapturing those data points, the data becomes clean (Fig 4.4). One data point deviates from most of the data points in fig 4.3, but it was not identified as an outlier by [OPERA](#) in the case of [SRS](#).

After following the [SRS](#) method, the [FPR](#) was recorded, as shown in Table 4.1.

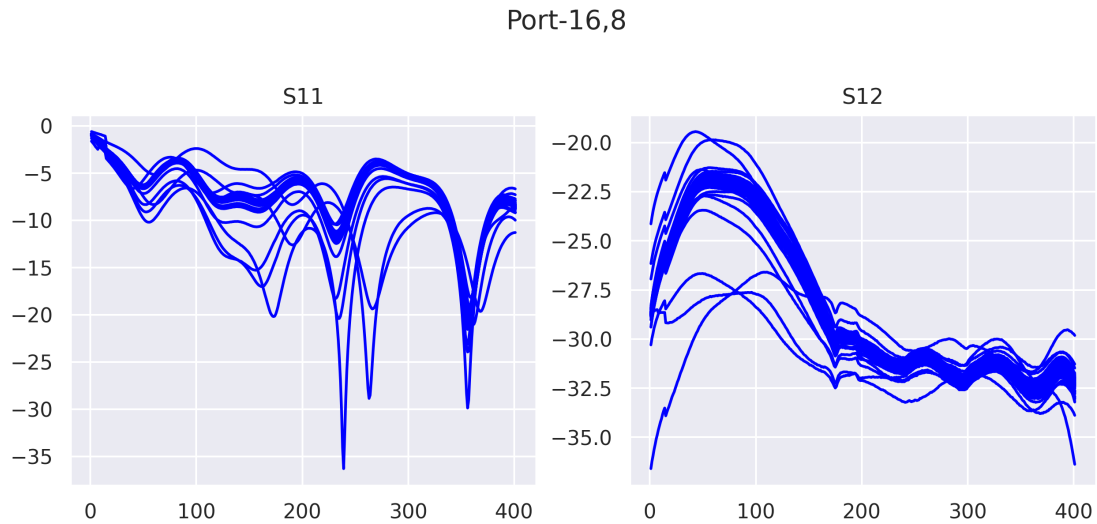


Figure 4.2: Data Captured by the user

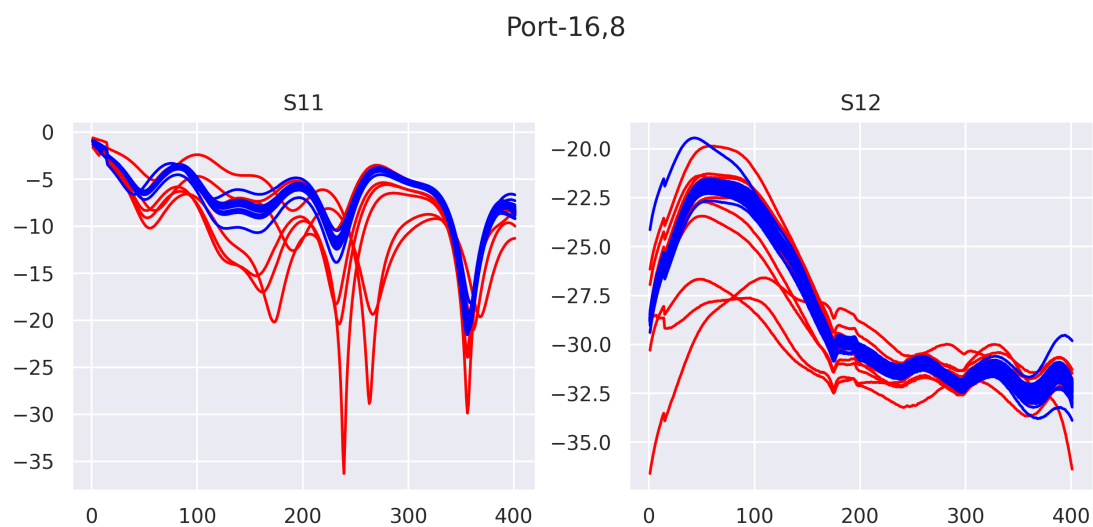


Figure 4.3: Miscaptures identified by OPERA

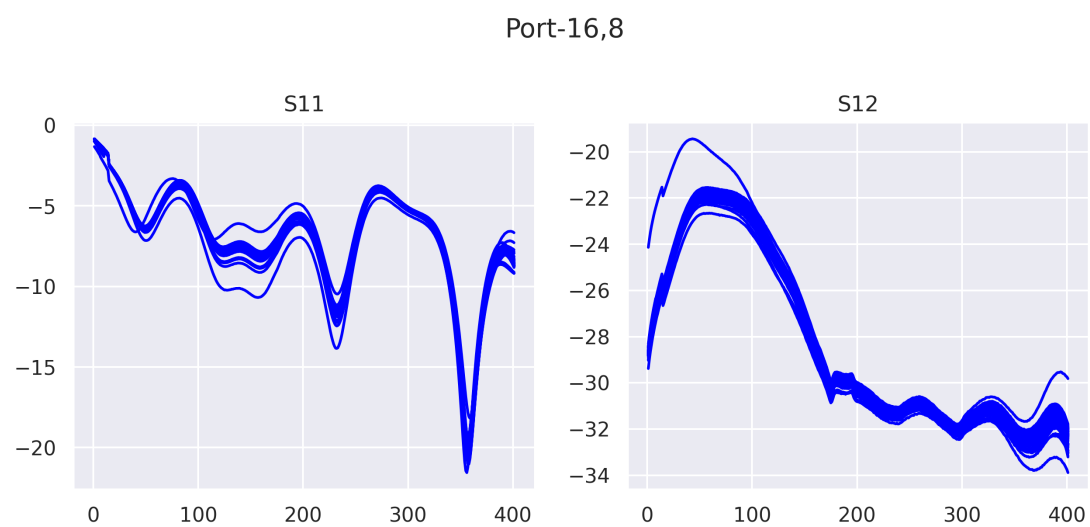


Figure 4.4: Data after recaptures

4.2.2 RSCM based Dataset

This follows a similar process in [SCM](#), with [OPERA](#) in place once the data capture session is complete.

We will allow the user to capture all the chips the way the user does in [SCM](#), without any change in the process. In [RSCM](#), all the data points are used as input to [OPERA.fit](#) method and the output of the method is shown in Figure 4.5.

In Figure 4.6, we see that the algorithm marks the [miscaptures](#) (in red) in the dataset. On careful inspection, it is visible that while using [RSCM](#), we can identify one extra data point as a miscapture, which was not identified by [SRS](#) (Figure 4.3).

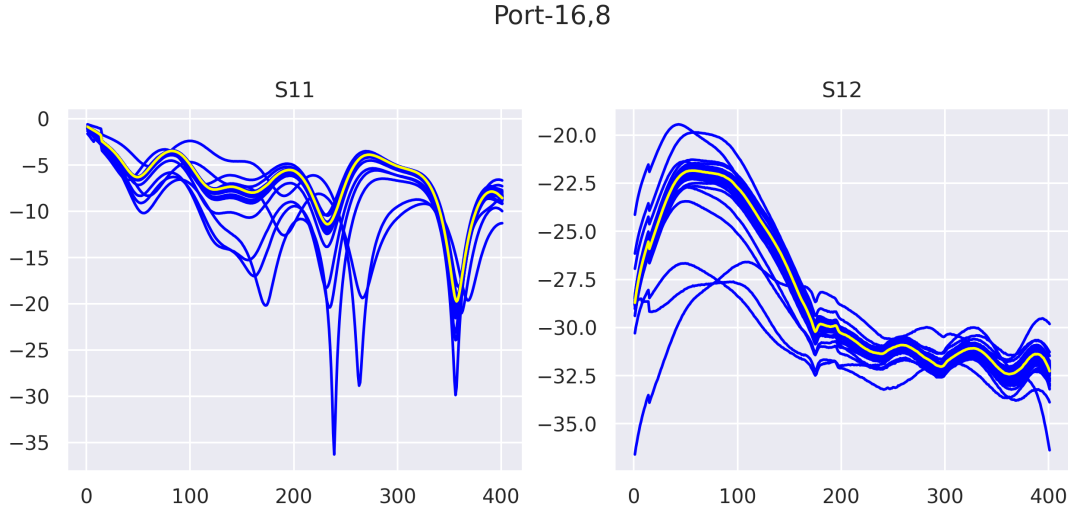


Figure 4.5: Data captured by the user with the output of [OPERA.fit](#) method (yellow)

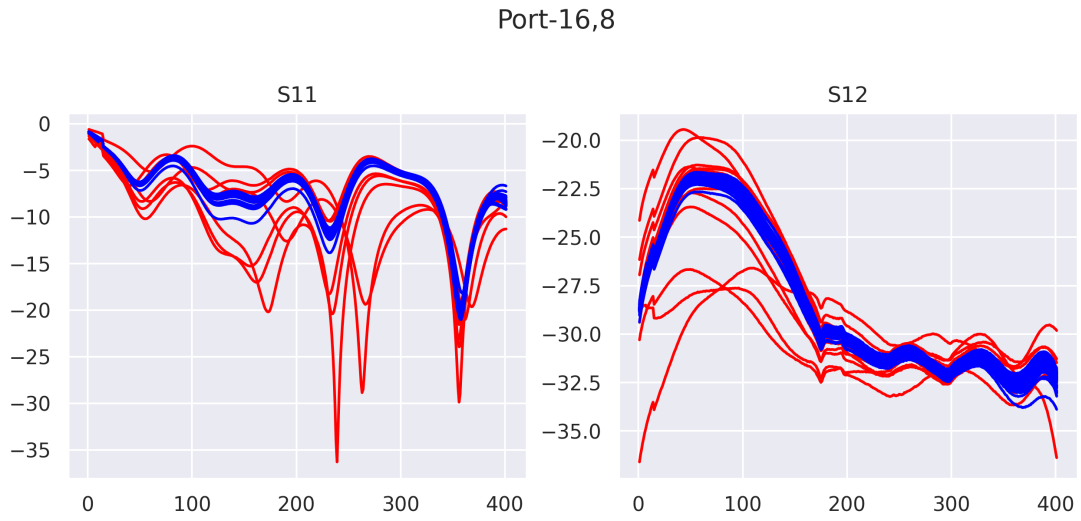


Figure 4.6: Miscaptures identified by OPERA

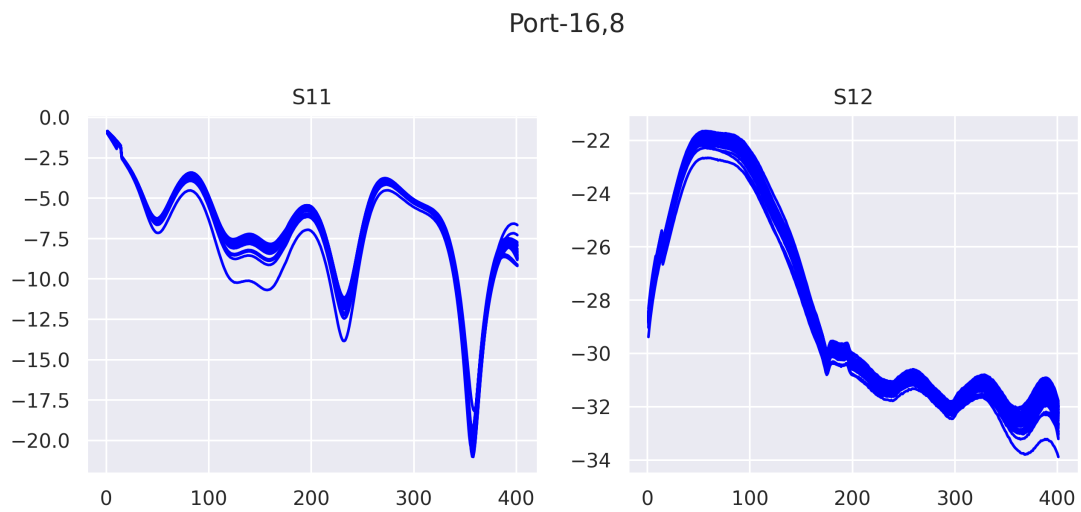


Figure 4.7: Data after recaptures

Once the [RSCM](#) is complete, we record the [FPR](#), as shown in Table 4.1, and on comparing it with the results in Table 4.1, there is a decrease in the [FPR](#) value. It is expected because of one extra miscapture identification by [RSCM](#), which is visible in Figure 4.6.

Method	False Positive Rate
SRS	0.020
RSCM	0.013

Table 4.1: False Positive Rate values using different data capture strategies

4.3 Case Study: [ECG](#)

As discussed in section 3.2.2, we will contaminate our dataset with abnormal [ECG](#) data and figure 4.9 shows the contaminated dataset that we will be using, with blue lines as normal [ECG](#) data and red lines as abnormal [ECG](#) data. Please note that we are contaminating the data to simulate real-life scenarios where some issues can happen while capturing the data.

Below are the steps followed to create this contaminated dataset.

- We randomly pick some data from the set of abnormal data, and we randomly replace the normal data entries with these selected abnormal data.
- The subset of abnormal data, from which we do the random selection for contamination, is never used in the testing data.

4.3.1 [SRS](#) based Dataset

Recalling [SRS](#) requires 2 steps: **Preparing a good representation of data and Data Quality Assurance**. Since the data has already been collected and is publicly available for this case study, we cannot perform step 1, where we will do repeated data captures. For our experiment, we will randomly select 5 data points from the normal [ECG](#) data and treat them as the good representation of the data. Figure 4.8 shows the randomly chosen data and the output of [OPERA](#).fit method (median).

Following this, the computed median is used to calculate the distance of all other data points in the dataset from the median, and the outliers are identified by [OPERA](#). The user

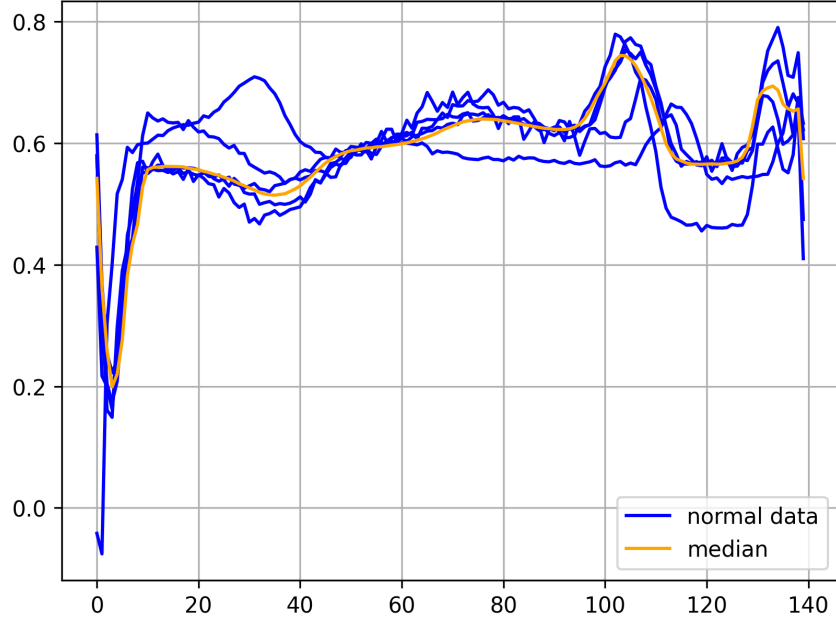


Figure 4.8: Randomly selected normal data and its median (representative)

is then asked to recapture those data points. Since we already have the correctly captured data points (recall that we replaced some of the normal data with abnormal to simulate the real-life scenario), we use that data to simulate the user behavior and record the [FPR](#). The false positives would then be genuine outliers in the dataset, not any [miscaptures](#).

4.3.2 [RSCM](#) based Dataset

In [RSCM](#), we let the user capture data for all the [targets](#) (similar to [SCM](#)) and then calculate the median using [OPERA](#).fit method. Then, we again incorporate the same procedure as performed by [OPERA](#).predict method, where we calculate the distance of each data point from this median, plot their [KDE](#), assign peaks and calculate [miscaptures](#). The user is asked to recapture the identified [miscaptures](#). Assuming clean data after the recapture, we can compute the [FPR](#), similar to what was done in [SRS](#).

Figure 4.10 shows the [miscaptures](#) identified by [OPERA](#), and Fig 4.11 shows the data after the recaptures are done. We can see two data entries in red, which [OPERA](#) again identifies as an outlier, which is true because they look different from other [targets](#).

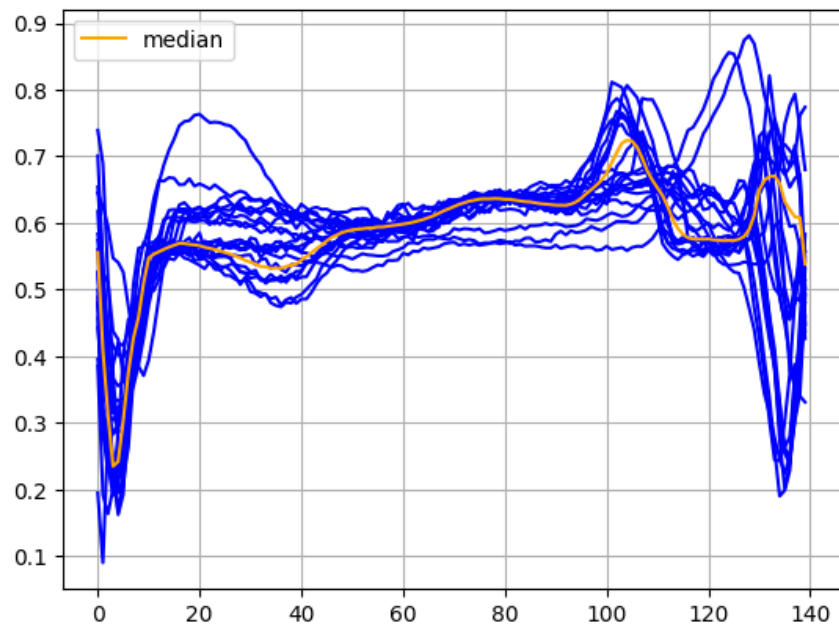


Figure 4.9: Data captured by the user and output of [OPERA.fit](#) method (yellow)

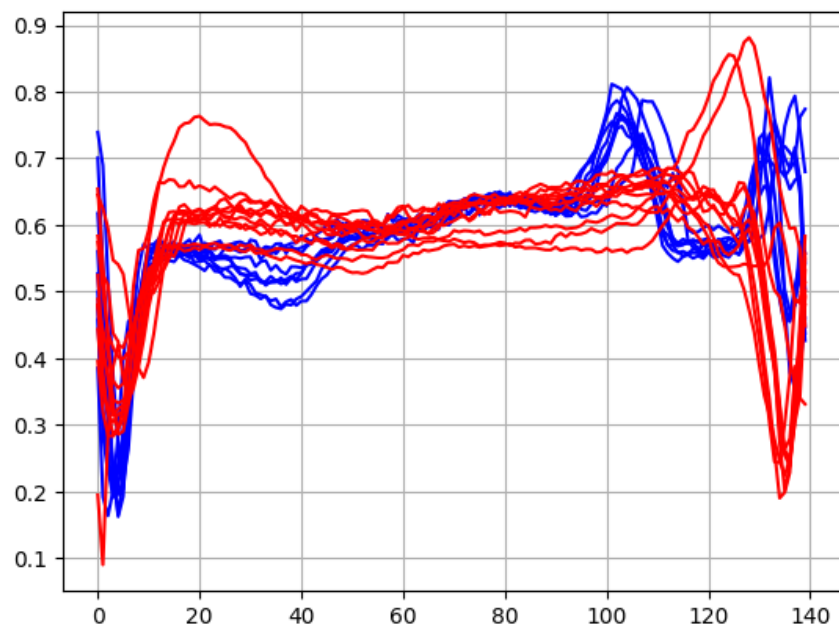


Figure 4.10: [miscaptures](#) identified by [OPERA](#)

Please note that since this is a public dataset and the data is already collected, we cannot actually do the data recapture process. We are trying to simulate the real-life scenario by replacing the correctly captured data with some abnormal data points that were kept separate at the start. To generate Fig 4.11, we are again replacing those identified *miscaptures* by *OPERA* with the correct data.

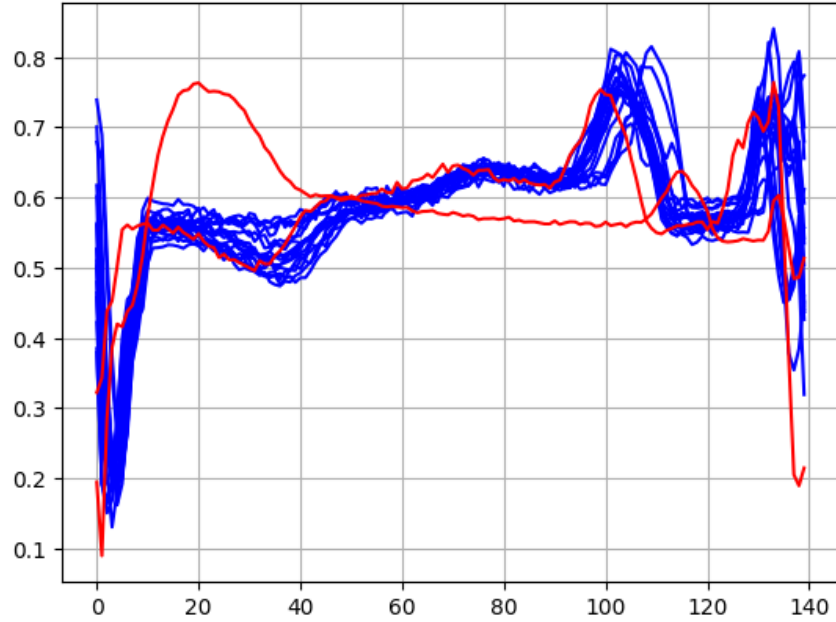


Figure 4.11: Data after recaptures

Method	False Positive Rate
SRS	0.058
RSCM	0.055

Table 4.2: *FPR* values using different data capture strategies

4.4 Comparison between strategies

This section will see how these strategies impact the *ML* models using the discussed datasets, to predict anomalies. To keep the analysis simple and straightforward, we will

try to keep a few things constant across comparisons. For this comparison, we will use OPERA as an anomaly detection algorithm instead of using it as just an outlier detection algorithm for *miscaptures*. Since OPERA does not require any parameters, this underlying algorithm would be the same in the upcoming results.

How to use OPERA as an Anomaly Detector?

As we have discussed in the previous sections, we are using OPERA to detect *miscaptures*. The term ‘Anomaly detection’ is often used interchangeably with outlier detection and depends on use case to use case. If we have a train set of normal data (Γ_{train}) and a test set containing normal and abnormal data (Γ_{test}), then we can use OPERA as an anomaly detector:

- Use Γ_{train} as an input to OPERA.fit method to calculate the median.
- Use Γ_{test} (normal + abnormal) as an input to OPERA.predict method to calculate the outliers.
- Since the median was calculated using normal data, the data points corresponding to abnormal data on the plot should be far away from the first peak and be identified as an anomaly.
- Since the data is already labeled, we can compute its FPR, FNR, and F1-score.

We will contaminate our training dataset with $\mu\%$ of *miscaptures* and record the performance of the Anomaly Detector. The model is evaluated on test data, which is kept separate from the training data and has no *miscaptures*. The test data contains a combination of normal data and anomalous data. Here, anomalous data are not *miscaptures*, but we know the ground truth for these anomalous data points that these are indeed anomalous.

If OPERA says a data point in the test dataset is an outlier, it would be called an anomalous data point. Similarly, if OPERA says it is not an outlier, it is predicted as normal. These predictions are then compared to the original labels in the test dataset, and we calculate the confusion metrics, which helps us to calculate the FPR, FNR, and F1-score.

We record this FPR, FNR, and F1-score for each value of μ , the percentage of *miscaptures* in the training data. Then, we plot the values of the F1-score on each μ value for each data capture method for comparison.

All three discussed data capture strategies take different amounts of time. Hence, we will also list the time taken for each process, which would help to make an informed decision about which method to use based on the criticality and priority of the data capture task.

4.4.1 Case Study: FTDI

For the FTDI dataset, we collected the data at ESG Lab on one of the ANVIL Checkpoints. After the complete data capture process, we found around 13 out of 74 FTDI chips were miscaptured because their data looked different from the other 61 FTDI chips. To perform our analysis here, we will split the dataset into train, test, and miscapture datasets. We will modify the train dataset by changing the μ value. For $\mu = 10\%$, we make sure that 10% of the data in the train dataset are miscaptures that we pick from the miscapture dataset. We do this procedure till $\mu = 50\%$. The test dataset contains a combination of nominal and anomalous FTDI chip data. To keep things simple, we ensure there are no miscaptures in the test dataset.

Figure 4.12 shows the change in the F1-score with different μ values, using each data capture method discussed in the thesis. When no data capture verification method exists, we can see the change in the F1-score for SCM. When the μ value increases, the value of the F1-score starts decreasing, which should be the case because the training data is now getting contaminated with miscaptures. This trend makes the model untrustworthy because when the μ value reaches 50% , the F1-score drops significantly, showing that the model has started learning incorrect patterns. Table 4.3 shows the FPR, FNR, and F1-score values for different μ values. The anomalous FTDI chips data is very far from the nominal chips and the miscaptures, due to which none of those anomalous chips are misclassified as nominal, and hence we always have the FNR values to be 0.

μ (%)	FPR	FNR	F1-Score
0	0.4359	0.0000	0.8903
10	0.5789	0.0000	0.8625
20	0.5500	0.0000	0.8625
30	0.5122	0.0000	0.8679
40	0.4884	0.0000	0.8679
50	0.9091	0.0000	0.7753

Table 4.3: Performance Metrics for Miscapture Rate using SCM

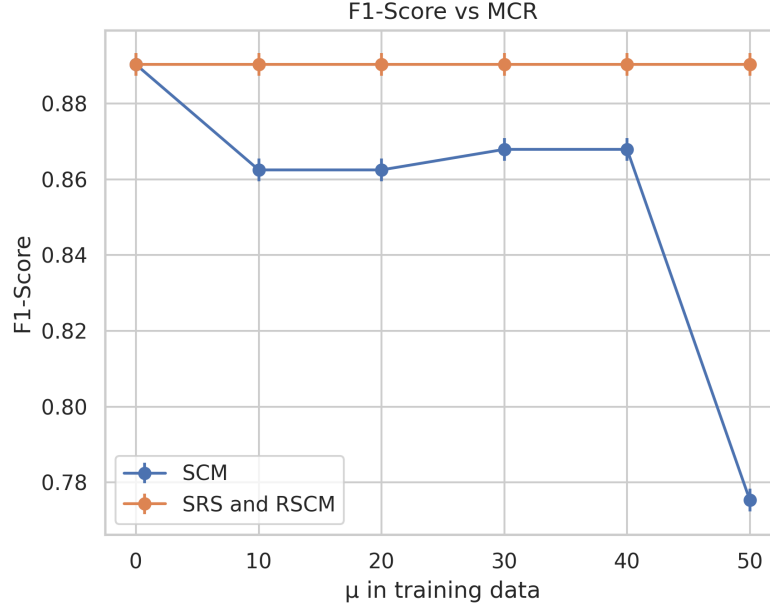


Figure 4.12: Change in F1-Score with increase in the values of μ for the 3 methods

On the contrary, if [SRS](#) or [RSCM](#) is used while capturing data, regardless of the μ value, the F1-score would always be the same because the user recaptures the [miscaptures](#) identified in the data capture process. However, these data recaptures increase the time taken to complete the data capture process and build the [ML](#) model. Table 4.4 shows the time all three methods take at different μ values. Assuming that the time taken to capture one [FTDI](#) chip on one of the ANVIL checkpoints is 1 minute, it is quite evident that [SCM](#) takes the least time as compared to other methods because of no requirement to recapture the data (at the expense of the model's performance). For [SRS](#) and [RSCM](#), the only difference in the time comes up at the start of the process. This is because we do multiple chip recaptures in [SRS](#), which is not the case in [RSCM](#). After that step, the time taken to complete the data captures would, on average, be similar to each other.

4.4.2 Case Study: [ECG](#)

Since this data is publicly available, we can only simulate the behavior of [miscaptures](#) to see the results. For this case study, we first separate the dataset into 3 parts: train, test, and unused. All data points in the train dataset are nominal, the test dataset contains a mixture of nominal and anomalous data points, and all data points in the unused dataset

μ (%)	SCM	SRS	RSCM
0	74	99	74
10	74	107.14	82.14
20	74	116.76	91.76
30	74	127.86	102.86
40	74	140.44	115.44
50	74	154.5	129.5

Table 4.4: Time to Model (in seconds) vs. μ values

are anomalous.

To simulate the [miscaptures](#) in the train dataset, we will use this unused dataset to contaminate the train dataset. With different values of μ in the train dataset, below is the performance of [OPERA](#), when used as an Anomaly Detector.

Figure 4.13 shows the change in the F1-score with an increasing value of μ for all three methods. When we increase the contamination in the training dataset, we can see that the model’s F1-score on the test dataset starts decreasing. This is expected because since the data quality is compromised in the train dataset, this might result in the model accepting [FP](#) or [FN](#). The [FPR](#), [FNR](#), and F1-score values are listed in Table 4.5.

When μ values go from 0 to 10, both [FPR](#) and [FNR](#) increase, with a larger increase in [FPR](#) than in [FNR](#). This trend is followed till μ value of 40. But after that, there is a drastic reduction of [FPR](#) but a huge increase in the [FNR](#) value, which results in a steep decline in the F1-score, as visible in the plot. When the μ value reaches 50, then 50% of the data has [miscaptures](#) in it, so the model is now predicting arbitrary patterns as nominal data, which resulted in this huge increase in [FNR](#). On the other hand, because the model has started to include these arbitrary patterns, the [FPR](#) decreased. This makes the model untrustworthy, which raises the concern and the requirement of having a streamlined data capture verification process.

On the contrary, the orange line shows the values of the F1-score on the test dataset when [SRS](#) or [RSCM](#) is deployed while capturing data. In this case, whatever the value of μ , since the user will always be asked to recapture it again, the [miscaptures](#) won’t be there at the end of the data capture process, and hence the F1-score would be, on average, similar for all μ values.

Another notable difference between these 3 approaches is the time taken by the user for data capture. If there is no data capture process in place ([SCM](#)), we won’t ask the

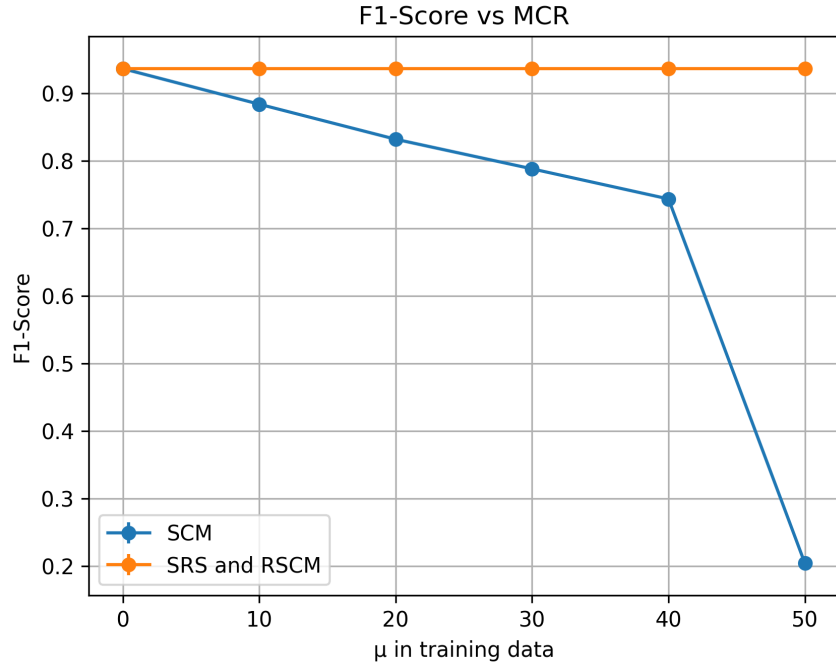


Figure 4.13: Change in F1-Score with increase in the values of μ for the 3 methods

μ (%)	FPR	FNR	F1-Score
0	0.0786	0.0230	0.9362
10	0.1654	0.0249	0.8836
20	0.2597	0.0288	0.8317
30	0.3418	0.0384	0.7879
40	0.4299	0.0518	0.7433
50	0.0636	0.8763	0.2040

Table 4.5: Performance Metrics for Miscapture Rate using [SCM](#)

user to recapture any [targets](#). So whatever the value of μ in the training dataset, the time taken to capture that training dataset would be constant. In the case of [SRS](#), we would first ask the user to capture the [ECG](#) data multiple times (5 in our case) on one target and do this process for randomly selected 5 [targets](#), as per the [SRS](#) process to calculate the good representation of the data. This would result in an increase in the data capture time. After this step, we would ask the user to capture the data again if our method flags any of the data captured as an outlier. Similar behavior would be there in the case of [RSCM](#), but here, the major difference is that we won't be doing multiple captures of the [ECG](#) in the first stage. Hence we will save a lot of time.

Table 4.6 compares the time taken to build the model with different μ values for each method. Assuming that the time taken to do 1 [ECG](#) capture for a patient is 5 minutes, we plot the values in the table. As explained above, it is clearly visible that [SCM](#) takes the least time and is not dependent on μ values, but the model that gets built with this data highly depends on μ values. The time taken by [RSCM](#) is slightly less than that taken by [SRS](#), and that is only because of the simplification of step 1 in [RSCM](#) by not asking the user to do multiple captures at the start.

μ (%)	SCM	SRS	RSCM
0	5000	5125	5000
10	5000	5675	5550
20	5000	6325	6200
30	5000	7075	6950
40	5000	7925	7800
50	5000	8875	8750

Table 4.6: Time to Model (in seconds) vs. μ values

Chapter 5

Conclusion

The success of [ML](#) models depends on the data on which they are trained. With this thesis, we touch base with issues in the data capture processes, which are usually considered unimportant. We introduced [OPERA](#), a parameterless and robust outlier detection algorithm, to improve the data quality well before it reaches the [ML](#) teams for further analysis.

We also introduced two approaches, namely [SRS](#) and [RSCM](#), and we applied them to two real-world datasets ([FTDI](#) chips and [ECG](#)). The results that we received with these 2 datasets give us a good validation of the discussed processes and their useability and benefits. With the above observations and results, we can say that [RSCM](#) gives us better results than the other methods.

In section 4.4, we used [OPERA](#) as an Outlier detection model and we try to contaminate the training data with miscaptures to see how a [ML](#) model ([OPERA](#) in this case) performs in predicting anomalies. The main idea is to use [OPERA](#) in the data capture process only, but in this case we are using it as a ML model on both our case studies, to have a fair comparison (by keeping the ML model same). Here, we leave the decision to the user to choose which data capture process is appropriate for them, considering the trade-off between time and performance. This decision would depend on various factors, like the criticality of the task or the priority of the task. The higher the priority of the task, less should be the less time taken, or the higher the criticality, the higher should be the performance of the model.

5.0.1 Future Work

In section 3.3, we introduced [OPERA](#), a parameterless outlier detection algorithm to detect outliers in a dataset. Currently, [OPERA](#) operates on univariate time series data. At the ‘cluster identification and assignments’ step, we mark all data points that don’t belong to the leftmost cluster as outliers. In the case of multivariate time series data, there can be multiple clusters with similar densities to the leftmost one, and the data points assigned to the clusters other than those should be marked as outliers. Hence, we suggest that [OPERA](#) can be extended to handle these multiple high-density clusters with densities similar to the leftmost one.

More investigation can be done on other available public datasets to verify the data capture verification approaches discussed in the thesis. This approach can have many applications. Consider a temperature sensor that records temperature every minute; suddenly, the sensor breaks or starts malfunctioning. It would result in a change in pattern which would be immediately reported to the user, and the issue with the sensor could be identified and fixed fast. Similar to this example, the approach can be further used in safety-critical systems like Autonomous driving cars. If a sensor captures some time series-related data, a significant change in the pattern due to some malfunctioning would be identified by [OPERA](#), raising the alarm to the passengers regarding the problem and suggesting relevant actions.

In the end, we would like to say that data quality is of immense importance in today’s world. It would help in making informed, unbiased, and confident decisions. More research and innovation should be done in this field to tackle the challenges that come with increased data-dependent processes.

References

- [1] Kernel density estimation - statsmodels. https://www.statsmodels.org/dev/examples/notebooks/generated/kernel_density.html.
- [2] *Central Limit Theorem*, pages 66–68. Springer New York, New York, NY, 2008.
- [3] Kernel density estimation - julien beaulieu’s gitbook, 2019. <https://julienbeaulieu.gitbook.io/wiki/sciences/programming/data-analysis/data-visualization/univariate-exploration-of-data/kernel-density-estimation>.
- [4] Kernel density estimation - ekamperi’s gitbook, 2020. <https://ekamperi.github.io/math/2020/12/08/kernel-density-estimation.html>.
- [5] Outlier detection methods in machine learning, 2021. <https://towardsdatascience.com/outlier-detection-methods-in-machine-learning-1c8b7cca6cb8>.
- [6] Kernel density estimation - deep ai, Accessed 2024. <https://deepai.org/machine-learning-glossary-and-terms/kernel-density-estimation>.
- [7] A Baranchuk, J Kang, et al. Electromagnetic interference of communication devices on ecg machines. *Clin Cardiol*, 32(10):588–592, Oct 2009.
- [8] Ben Lutkevich. Sensor data, 2023. <https://www.techtarget.com/iotagenda/definition/sensor-data>.
- [9] Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. *The Annals of Statistics*, 38(5), October 2010.
- [10] T Buczkowski, D Janusek, et al. Influence of mobile phones on the quality of ecg signal acquired by medical devices. *Measurement Science Review*, 13(5):231–236, 2013.

- [11] Censius AI. Importance of data quality. Website, 2022. <https://censius.ai/blogs/importance-of-data-quality>.
- [12] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*. Springer, Boston, MA, 2016.
- [13] Yan Chen, Yan Hao, Thanawin Rakthanmanon, et al. A general framework for never-ending learning from time series streams. *Data Mining and Knowledge Discovery*, 29:1622–1664, 2015.
- [14] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. Autoencoder-based network anomaly detection. In *2018 Wireless Telecommunications Symposium (WTS)*, pages 1–5, 2018.
- [15] Mahmoud A Dakka, Thanh V Nguyen, James M M Hall, et al. Automated detection of poor-quality data: case studies in healthcare. *Scientific Reports*, 11(1):18005, 2021.
- [16] Zhiyi Zhang Douglas S. Shafer. *Introductory Statistics*. Saylor Foundation, 2012.
- [17] Thorben Finke, Michael Krämer, Alessandro Morandini, Alexander Mück, and Ivan Oleksiyuk. Autoencoders for unsupervised anomaly detection in high energy physics. *Journal of High Energy Physics*, 2021(6), June 2021.
- [18] FTDIChip. Ft232rl, 2024. <https://ftdichip.com/products/ft232rl/>.
- [19] K S Gellert, P Rautaharju, M L Snyder, E A Whitsel, K Matsushita, G Heiss, and E Z Soliman. Short-term repeatability of electrocardiographic tpeak-tend and qt intervals. *Journal of electrocardiology*, 47(3):356–361, 2014.
- [20] Global News. Amazon job recruiting tool 'biased against women', 2018. <https://globalnews.ca/news/4532172/amazon-jobs-ai-bias/>.
- [21] Ary L Goldberger, Luis A N Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen C Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23):E215–20, Jun 2000.
- [22] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and

- H Eugene Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.
- [23] Vinutha H P, B. Poornima, and B. Sagar. *Detection of Outliers Using Interquartile Range Technique from Intrusion Dataset*, pages 511–518. 01 2018.
 - [24] Tao Huang, Cynthia A James, Crystal Tichnell, Brittney Murray, Jingchuan Xue, Hugh Calkins, and Larisa G Tereshchenko. Statistical evaluation of reproducibility of automated ecg measurements: an example from arrhythmogenic right ventricular dysplasia/cardiomyopathy clinic. *Biomedical signal processing and control*, 13:23–30, 2014.
 - [25] Mia Hubert, Michiel Debruyne, and Peter J. Rousseeuw. Minimum covariance determinant and extensions. *WIREs Computational Statistics*, 10(3), December 2017.
 - [26] Oleg Iegorov and Sebastian Fischmeister. Parameterless semi-supervised anomaly detection in univariate time series. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*, page 644–659, Berlin, Heidelberg, 2020. Springer-Verlag.
 - [27] InfluentialPoints. Coefficient of variation: Use and misuse. https://influentialpoints.com/Training/Coefficient_of_variation_Use_and_misuse.
 - [28] Johns Hopkins. Event monitor. <https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/event-monitor>.
 - [29] Kerri Hale. What to do when bad data thwarts machine learning success, 2018. <https://towardsdatascience.com/what-to-do-when-bad-data-thwarts-machine-learning-success-fb82249aae8b>.
 - [30] K Khunti. Accurate interpretation of the 12-lead ecg electrode placement: A systematic review. *Health Education Journal*, 73(5):610–623, 2013.
 - [31] et al. Kligfield. Recommendations for the standardization and interpretation of the electrocardiogram part i: The electrocardiogram and its technology. *Circulation*, 115:1306–1324, 2007.
 - [32] LinkedIn. What are the most common data capture errors and how to avoid them. <https://www.linkedin.com/advice/0/what-most-common-data-capture-errors-how-avoid>.

- [33] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.
- [34] Andreea Roxana Luca, Tudor Florin Ursuleanu, Liliana Gheorghe, Roxana Grigorovici, Stefan Iancu, Maria Hlusneac, and Alexandru Grigorovici. Impact of quality, type and volume of data used by deep learning models in the analysis of medical images. *Informatics in Medicine Unlocked*, 29:100911, 2022.
- [35] Mayo Clinic. Electrocardiogram (EKG or ECG): About, 2022. <https://www.mayoclinic.org/tests-procedures/ekg/about/pac-20384983>.
- [36] K McCann, A Holdgate, R Mahammad, and A Waddington. Accuracy of ecg electrode placement by emergency department clinicians. *Emerg Med Australas*, 19(5):442–448, Oct 2007.
- [37] Carlos Moreno, Sebastian Fischmeister, and Philippe Vibien. A method and apparatus for detection of counterfeit parts, compromised or tampered components or devices, tampered systems such as local communication networks, and for secure identification of components.
- [38] NORAV Medical. Exploring 5 different types of ECG machines: A comprehensive overview. <https://www.noravmedical.com/exploring-5-different-types-of-ecg-machines-a-comprehensive-overview/>.
- [39] NORAV Medical. Resting ecg system solution. <https://www.noravmedical.com/categories-products/resting-ecg/>.
- [40] Palitronica. Anvil, 2024. <https://www.palitronica.com/products/anvil>.
- [41] M.A. Peberdy and J.P. Ornato. Recognition of electrocardiographic lead misplacements. *Am J Emerg Med*, 11:4034, 1993.
- [42] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.
- [43] Towards Data Science. Data quality issues that kill your machine learning models, 2023. <https://towardsdatascience.com/data-quality-issues-that-kill-your-machine-learning-models-961591340b40>.

- [44] Valerie Sessions and Marco Valtorta. The effects of data quality on machine learning algorithms. pages 485–498, 01 2006.
- [45] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Routledge, 1st edition, 1998.
- [46] Simplilearn. Probability density function, Accessed 2024. <https://www.simplilearn.com/tutorials/statistics-tutorial/probability-density-function>.
- [47] Saket Srivastava, Rushit N. Shah, Catalin Teodoriu, and Aditya Sharma. Impact of data quality on supervised machine learning: Case study on drilling vibrations. *Journal of Petroleum Science and Engineering*, 219:111058, 2022.
- [48] TASQ.AI. Data collection challenges in ai. Website, 2022. <https://www.tasq.ai/blog/data-collection-challenges-in-ai/>.
- [49] Hossein Torabi, Seyyed Lavasani Mirtaheri, and Salvatore Greco. Practical autoencoder based anomaly detection by using vector reconstruction error. *Cybersecurity*, 6(1), 2023.
- [50] Philippe Vibien. SilGeo: A method for the detection of counterfeit, compromised, or tampered electronic devices. Master’s thesis, University of Waterloo, August 2022.

Glossary

ANVIL The product version of SilGeo by Palitronica Inc. [iii](#), [ix](#), [6](#), [7](#), [15](#), [16](#), [27](#), [44](#)

miscaptures Miscaptures are the data entries which have not been correctly captured in the data capture process [x](#), [1](#), [2](#), [16](#), [22](#), [28](#), [29](#), [32](#), [34](#), [35](#), [37](#), [40–46](#)

SilGeo A system for creating hardware fingerprints based on impedance discontinuities and wave propagation characteristics [5](#), [6](#)

targets Targets can be any subject or entity for which the data collection is intended. [13–15](#), [23](#), [25](#), [28](#), [29](#), [34](#), [40](#), [48](#)