



**MASTER Thesis**

**Preprocessing Machine  
Learning Input Data with a  
Kalman Filter**

**Submitted by:**  
Aiman Zehra

**Matriculation number**  
1388996

**Under the guidance of**  
Prof. Dr. Andreas Pech  
Prof. Dr. Peter Nauth

# Declaration

I, Aiman Zehra, confirm that I have written this thesis titled “Preprocessing machine learning input data with a Kalman filter” on my own under the guidance of Prof. Dr. Andreas Pech and Prof. Dr. Peter Nauth. No other sources were used except those referenced. Content which is taken literally or analogously from published or unpublished sources is identified as such. The drawings or figures of this work have been created by me or are provided with an appropriate reference. This work has not been submitted in the same or similar form to any other examination board.



**Aiman Zehra**

November 15, 2024

# Abstract

The aim of this study is to optimize a classifier for person detection in an office scenario. The objective is to improve a machine learning model—such as Convolutional Neural Networks (CNN), Multi-Layer Perceptrons (MLP), or Support Vector Machines (SVM)—that can reliably distinguish between occupied and unoccupied office seats using ultrasonic reflections by using a Kalman filter. The Kalman filter is used here for building a model for estimating the sound pressure and noise suppression, thus improving the classification accuracy.

The experimental setup employs the use of ultrasonic sensor ‘Red Pitaya’ for the collection of datasets in case of occupancy and non-occupancy of office chairs. Within this thesis, an extensive comparative analysis is conducted with as well as without Kalman filter to demonstrate the effectiveness of Kalman filter in the improvement of classification.

The machine learning model developed, is trained with significant number of datasets for the classification. Different performance metrics such as accuracy, precision, recall and F1-score show marked improvements when the Kalman filter is applied.

Furthermore, the findings of this thesis study demonstrate the practical benefits of incorporating the Kalman filter with the advanced machine learning model that can broadly contribute towards the smart office technology offering efficient occupancy detection system.

**Keywords** – *Ultrasonic Sensor, Red Pitaya, CNN, MLP, SVM, Kalman Filter, Performance metric, Accuracy, Precision, Recall, F1- score, Smart Office Technology, Machine Learning*

# Acknowledgement

I would sincerely like to thank my thesis supervisors Prof. Dr. Andreas Pech and Prof. Dr. Peter Nauth for providing me full support through the entire journey of this thesis. The apt technical support and guidance, uncomplicated knowledge transfer and regular feedback were a big help to gain the needed knowledge and include the same to improve my thesis work. I would once again like to thank Prof. Dr. Andreas Pech for being patient enough to guide me throughout the research work. Moreover, I would also like to extend my earnest gratitude to my family and friends for being helpful and supportive during the hard times.

Finally, I would like to thank each and every one who supported and guided me in any way, during my Masters in Information Technology, at Frankfurt University of Applied Sciences.

Sincerely,  
**Aiman Zehra**

# Contents

|                                                        |               |
|--------------------------------------------------------|---------------|
| <b>Introduction.....</b>                               | <b>12</b>     |
| 1.1. Motivation and Objective.....                     | 12            |
| 1.2. Problem Statement.....                            | 13            |
| 1.3. Scope of Study .....                              | 13            |
| 1.4. Structure of the Document.....                    | 14            |
| <br><b>Theoretical Background.....</b>                 | <br><b>16</b> |
| 2.1. Introduction to Person Detection .....            | 16            |
| 2.2. Ultrasonic Sensors .....                          | 17            |
| 2.3. Principle of Ultrasonic Sensor.....               | 18            |
| 2.4. Ultrasonic Data Collection Software .....         | 19            |
| 2.5. Red Pitaya Sensor.....                            | 20            |
| 2.6. Machine Learning Approach.....                    | 21            |
| 2.7. Multi-Layer Perceptrons (MLP).....                | 23            |
| 2.8. Confusion Matrix .....                            | 24            |
| 2.9. Kalman Filter.....                                | 26            |
| 2.10. Thresholding Methods .....                       | 27            |
| 2.11. Signal Envelope Process .....                    | 28            |
| 2.12. Graphical User Interface (GUI).....              | 29            |
| <br><b>Literature Review .....</b>                     | <br><b>31</b> |
| 3.1. Occupancy Detection.....                          | 31            |
| 3.2. Ultrasonic Sensor Classification .....            | 32            |
| 3.3. Kalman Filter in Occupancy Detection Systems..... | 32            |
| 3.4. Machine Learning.....                             | 32            |
| 3.5. Feature Extraction and Thresholding Methods ..... | 33            |
| <br><b>Methodology .....</b>                           | <br><b>34</b> |
| 4.1. Objective .....                                   | 34            |
| 4.2. General Architecture of the system.....           | 34            |
| 4.3. Physical Experimental Work Setup.....             | 37            |

|                                 |                                                 |           |
|---------------------------------|-------------------------------------------------|-----------|
| 4.4.                            | Dataset Acquisition.....                        | 37        |
| 4.5.                            | Software Setup .....                            | 42        |
| 4.6.                            | Kalman Filter Model Development.....            | 45        |
| 4.7.                            | MLP Classification Model .....                  | 51        |
| <b>Result and Analysis.....</b> |                                                 | <b>56</b> |
| 5.1.                            | Noise reduction Analysis.....                   | 56        |
| 5.2.                            | Impact of Kalman Filter on Classification ..... | 59        |
| 5.3.                            | Comparative Performance Analysis.....           | 65        |
| 5.4.                            | Summary and Perspective .....                   | 66        |
| <b>References.....</b>          |                                                 | <b>68</b> |

# List of Figures

|                                                                                                                                                |    |
|------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1 SRF02 Ultrasonic Sensor (robot_electronics) .....                                                                                          | 17 |
| 2.2 Principle working of Ultrasonic Sensor (rolling robot, 2021) .....                                                                         | 18 |
| 2.3 GUI for capturing dataset .....                                                                                                            | 20 |
| 2.4 Red Pitaya with inbuild ultrasonic sensor .....                                                                                            | 21 |
| 2.5 ML model with testing and training phases (Aggarwal, 2023) .....                                                                           | 22 |
| 2.6 MLP with two hidden layers (Jaisawal) .....                                                                                                | 23 |
| 2.7 Confusion Matrix ( NB Share) .....                                                                                                         | 25 |
| 2.8 Signal and its corresponding envelope (Dehghan-Niri, Ehsan, Alireza, & Salvatore, 2013) .....                                              | 29 |
| 4.1 General Architecture of the System.....                                                                                                    | 34 |
| 4.2 Red Pitaya GUI for measurement .....                                                                                                       | 35 |
| 4.3 Experimental setup with empty office chair .....                                                                                           | 38 |
| 4.4 Experimental setup with a cardboard box kept on an office chair. ....                                                                      | 39 |
| 4.5 Experimental setup with a person sitting idle on an office chair. ....                                                                     | 40 |
| 4.6 Experimental setup with a person sitting on an office chair, continuously moving hands and multiple obstacles kept in nearby vicinity..... | 41 |
| 4.7 Raw dataset from the UDP Client .....                                                                                                      | 43 |
| 4.8 Flowchart of Kalman Filter Model .....                                                                                                     | 45 |
| 4.9 GUI for the Kalman Filter Process.....                                                                                                     | 49 |
| 4.10 GUI depicting Left plot shows the raw signal processing, right plot shows the processed signal after Kalman filtering.....                | 50 |
| 4.11 GUI showing the option to save the Niblack peak data after the Kalman Filter processing .....                                             | 50 |
| 4.12 GUI showing the option to save the Yanowitz peak data after the Kalman Filter processing.....                                             | 51 |
| 4.13 Flowchart of the MLP Classification Model.....                                                                                            | 51 |
| 5.1 Plot (a) and plot (b) showing the signal amplitude along with peaks and envelope for occupied dataset .....                                | 57 |
| 5.2 Plot (a) and plot (b) showing the signal amplitude along with peaks and envelope for occupied dataset .....                                | 58 |

5.3 Confusion matrix for case I .....60

5.4 Bar chart for the comparison of matrices for case I.....60

5.5 Confusion matrix for case II.....61

5.6 Bar chart for the comparison of matrices for case II .....62

5.7 Confusion matrix for case III .....63

5.8 Bar chart for the comparison of matrices for case III.....63

5.9 Confusion matrix for case IV .....64

5.10 Bar chart for the comparison of matrices for case IV .....65



# List of Tables

- 4.1 Detailed Headers Description.....43
- 5.1 Classification parameter table for non-occupied chair (empty) vs. occupied chair (Human Idle) .....59
- 5.2 Classification parameter table for non-occupied chair (Boxes) vs. occupied chair (Human Idle) .....61
- 5.3 Classification parameter table for non-occupied chair (Empty) vs. occupied chair (Human moving) .....62
- 5.4 Classification parameter table for non-occupied chair (Boxes) vs. occupied chair (Human moving) .....64

# Listings

|                                                                                                                    |    |
|--------------------------------------------------------------------------------------------------------------------|----|
| 4.1 Code snippet depicting the logic of removing the headers from the raw dataset.                                 | 44 |
| 4.2 Code showing the calculation of the estimated sound pressure.....                                              | 47 |
| 4.3 Code snippet showcasing the of Niblack and Yanowitz local thresholding.....                                    | 47 |
| 4.4 Code snippet with logic to detect Niblack and Bruckstein peak and store data                                   | 48 |
| 4.5 Code for signal envelope.....                                                                                  | 48 |
| 4.6 Code to load the input file .....                                                                              | 52 |
| 4.7 Code to save three output files with actual values, predicted values and<br>performance result of matrix. .... | 52 |
| 4.8 Code logic with the calculation of specificity, recall and precision.....                                      | 53 |
| 4.9 Code with the logic of splitting the data into testing and training sets. ....                                 | 53 |
| 4.10 Code logic for normalization .....                                                                            | 54 |
| 4.11 Code logic with the maximum iterations.....                                                                   | 54 |
| 4.12 Code for confusion matrix and other classification parameters .....                                           | 55 |

# Abbreviations

**AI:** Artificial Intelligence  
**AM:** Amplitude Modulation  
**ADC:** Analog to Digital Converter  
**CNN:** Convolutional Neural Networks  
**DAC:** Digital to Analog Converter  
**DL:** Deep Learning  
**FFT:** Fast Fourier Transform  
**FN:** False Negative  
**FP:** False Positive  
**GUI:** Graphical User Interface  
**HVAC:** Heating, Ventilation Air Conditioning  
**IMU:** Inertial Measurement Unit  
**IR:** InfraRed  
**KF:** Kalman Filter  
**ML:** Machine Learning  
**MLP:** Multi-Layer Perception  
**RP:** Red Pitaya  
**ReLU:** Rectified Linear Unit  
**SVM:** Support Vector Machine  
**TP:** True Positive  
**TN:** True Negative  
**WSN:** Wireless Sensor Network

# Chapter 1

## Introduction

Human detection and Activity analysis play a pivotal role in variety of places, including human-computer interaction, security monitoring, assisted living, and person tracking, smart office buildings. Building occupancy data is highly desired to boost the efficiency of energy management systems. Occupancy information in devices can be provided with high precision using multiple technologies. Furthermore, accurate human detection in an office setting is made possible by ultrasonic sensors (Mittal, Mongia, & Chugh, 2022).

In an office setting, a machine learning model can be utilized to improve human detection accuracy. In scientific terms, machine learning (ML) is the study of statistical models and methods that computer systems use to do certain tasks without the need for explicit programming. It is one of the most rapidly expanding technical fields of today, lying at the confluence of computer science and statistics, as well as data science and artificial intelligence (AI). Recent breakthroughs in machine learning have been primarily driven by new learning algorithm theory and the ongoing expansion of low-cost processing and data availability (Aggarwal, 2023).

This thesis employs use of Kalman filter for an effective improvement of the machine learning model for precise human detection. Any undesired signal that may deteriorate quality and intelligibility is referred to as additive noise. The Kalman filter helps to effectively suppress this noise which can be used for further processing.

### 1.1. Motivation and Objective

When it comes to the efficient management of space and resources with the office environment for sustainable living and cost effectiveness, the precision of the ultrasonic sensor for accurate detection plays a crucial role. Incorrect estimations of building occupancy can occur in certain circumstances can lead to inefficient utilization of the resources.

The only differences are caused by incorrect estimations of building occupancy. This error rate between reality and prediction can be minimized by enhancing the accuracy of occupancy detection. There is much literature making claims that several research studies considered occupancy accordingly. Methods applied to support findings so far by different scientific publications still vary immensely from one to another. Different methodologies and academic papers have been published in the occupancy domain (Mittal, Mongia, & Chugh, 2022).

Precise building occupancy data is highly desirable for optimizing the

performance of energy management strategies. The precise provision of occupancy information in devices can be achieved using a multitude of ways. Furthermore, the environment has been programmed to use ultrasonic sensors like "Red Pitaya" (Mittal, Mongia, & Chugh, 2022).

Proper person detection within office spaces means we can better save energy, improve security, and create a more convenient environment for people to work in. Offices could optimize their energy usage with dynamic lighting per occupancy for potential savings. It ensures safety through precise detection, monitors restricted areas, and prevents unauthorized access by providing an alarm signal to the security process on accurate matching alerts in real-time ensuring heightened levels of overall protection making emergency evacuation more efficient. Moreover, for convenience, the person detection system might automatically adjust settings related to lighting and temperature to make the environment more comfortable to work in. In short, person detection will not only translate into energy savings and enhanced security but also increased user comfort in office buildings.

## 1.2. Problem Statement

The improvement of real-time office environment energy conservation, security and user emerge even better by deploying person detection system at appropriate in the workplace. But the existing detection approaches based on visual and infrared sensors typically involve setbacks in respect to privacy, costs, or performance under different environmental conditions. While ultrasonic sensors provide a solution that is non-intrusive and cost-effective, rendered data has high levels of noise which affects the quality and accuracy of human detection models.

By utilizing ultrasonic sensors and cutting-edge machine learning algorithms, this thesis seeks to solve the problem of optimizing a classifier for human detection in an office setting. It aims to improve the stability, accuracy of detection system by implementing Kalman filter for eliminating noise and incorporating with a machine learning model for occupied chair detection.

## 1.3. Scope of Study

This empirical study limits the scope discussing a classifier to detect people in an office environment which has been developed and optimized by means of ultrasonic sensor data along with machine learning techniques. The research addresses several important pillars:

- **Machine Learning Model:** The work will implement and evaluate machine learning model, specifically Convolutional Neural Network (CNN), Multi-Layer Perceptron (MLP) or Support Vector Machine (SVM) to classify ultrasonic data.
- **Kalman Filter:** The study proposes to examine the relevance of Kalman filter, the objective being to suppress unwanted "noise" of the detection system and calculate sound pressures in the context of modeling. Ultimately, such an adjustment may promote more valid machine learning models.

- **Performance Evaluation:** The classifiers will be evaluated in terms of its performance with and without using the Kalman filter. Results will be compared using metrics such as Accuracy, Precision, Recall and F1-score.
- **Controlled Office Environment:** The tests will be conducted in an office to have the experiments run consistently and reliably. There will be no mention of any other environments that could interfere with the detection system and thus this study.

## 1.4. Structure of the Document

The structure of the thesis is as follows:

**Chapter 2:** This chapter gives the theoretical background of different aspects of person detection using ultrasonic sensor. The basic principle of working for the ultrasonic sensor is also depicted in this section.

Furthermore, the detailed working of a Red Pitaya sensor as well as the machine learning approach along with different classification methods are also explained. For the efficient classification, different parameters including confusion matrix, F1-score, precision, recall etc. are also discussed.

**Chapter 3:** This chapter provides overview of previous research and methods regarding person detection, ultrasonic sensors, machine learning models as well as Kalman filters. It presents a detailed review of the most recent methodologies in literature and shortcomings within research that this thesis aims to improve. The chapter comprises several sections that introduce the underlying principle of ultrasonic sensor, classification, Machine learning algorithm, Kalman filter in occupancy detection, Feature extraction thresholding methods and previous works on combining these technologies.

**Chapter 4:** This chapter details a study methodology that includes the generation of machine learning models, incorporation of Kalman filter based algorithms for noise suppression, Sound pressure calculation as well as data acquisition with its preprocessing. For ensuring the scientific rigor and reliability of their results, it explains how experimentation was done, detailing each step involved.

Additionally, it also deals with the experimental setup of ultrasonic sensors on office chairs for data capture, labeling of gathered dataset and normalization. It also specifies Kalman filter for noise reduction and Evolution of CNN, MLP or SVM model. This section clearly emphasizes on both experimental methodologies for evaluating model performances as well as measuring these evaluations.

**Chapter 5:** This chapter marks a significant milestone revealing the findings of experiments conducted. This study evaluates machine learning models with and without the integration of Kalman filter by performance metrics including Accuracy, Precision and Recall and F1-score. Results are shown via graphs, tables, and matrix. This chapter presents in-depth analysis of classifier performance and reveals how use of Kalman filter has enhances the classification of occupied and unoccupied office chairs.

Furthermore, it encapsulates the statistical validation of the improvement in the machine learning model in terms of classification as well as discusses the robustness of different scenarios. It basically reveals the analysis and interpretation. The study focuses on noise suppression by Kalman filter, sound pressure calculation, and its potential improvement in the machine learning model performance for classification. Results are linked to literature review pointing out their relevance.

Moreover, this chapter considers the practical implications of the research by reviewing how an improved detection system could be implemented in real office environments. It also addresses any unexpected observations and gives an explanation. The discussion also considers scalability and future models of advanced system.

In Addition, it is summarized in this study about the highlights and improvements for Kalman filter integration with machine learning models for ultrasonic sensors. It also illustrates the opportunities to follow-on research, such as studying other sensor types or improving the model accuracy for different atmosphere. Each chapter flows nicely from the introduction to problem statement, presentation of findings and analysis right down to summarizing overall contribution of the study. The reader will find the research to be engaging and educational due to the systematic approach's certainty of a thorough and lucid presentation of the data.

## Chapter 2

# Theoretical Background

### 2.1. Introduction to Person Detection

Person detection and Monitoring activities have been used in a variety of settings, including assisted living, security, person tracking, and human-computer interface. In recent times, the swift advancements in computer vision and various network devices have made human behavior understanding, tracking, and person authentication indispensable for delivering customized services that meet users' explicit and implicit needs. In such a smart environment, face, voice, gait, individual trajectories, and other information are adopted for multiple person detection and real-time tracking. In the meanwhile, the system based on an ultrasonic sensor is a useful method for detecting occupied chairs in the office area with optimization in the design of a room and estimation of the capacity for daily life of an elderly person. (Tao, Kudo, Nonaka, & Toyama, 2012)

Moreover, Conventional authentication methods, which rely on different biometric evidence like fingerprint, iris, speech, and palm vein, can sustain a high degree of security, but user cooperation is required. Many research has also used cameras to analyze human behavior; nonetheless, there is a chance that cameras will infringe users' privacy to some degree. However, in reality, wrong identification of a user or recognition of activity is a great threat in day-to-day operations. Any physical or psychological issue must be taken into account (Tao, Kudo, Nonaka, & Toyama, 2012).

Many of the large-scale production facilities use surveillance systems like networks of several cameras for monitoring purposes. Ensuring security, safety, and quality-that is, according to established protocols to produce services-and preventing activities that may cause danger to people is the aim. Normally, this would be something for which a surveillance operator should monitor manually, which is subjective and ineffective, especially in the presence of several running video streams. The first steps toward automated monitoring include robust human detection using ultrasonic sensors. People identification based only on visual observations is a tremendously tricky task, especially within industrial environments. Within these types of situations, current techniques, including color background modeling and general-purpose person detectors, work appallingly. There are several reasons for this: occlusions, shifting objects, difficult backgrounds, and sparks and vibrations do not enable the distinguishing of people (Mörzinger, Thaler, Stalder, Grabner, & Van Gool, 2011).



- **Relevance to the office Environment**

Person detection is highly useful in office settings where the same principle may be used to regulate lighting and HVAC systems based on occupancy. This saves costs that relate directly to reducing energy usage, the consequences of which are smaller environmental effects. It is also a very critical component in security and access control to certain parts of the office. The present environment requires the development of an environment friendly yet secure area for the detection and tracking of people without intruding on the privacy of individuals. Person detection is itself a field of growth, and with the advancement in sensor technology and machine learning, it has been able to develop further. Integrating these, possibilities have never been higher to get smarter and responsive systems, particularly concerning smart buildings or office automation. This section has provided a relatively high level of introduction to person detection technologies in general as foundational work that will take us later to deeper discussions on ultrasonic sensors and machine learning techniques.

## 2.2. Ultrasonic Sensors

An ultrasonic sensor is an electronic device that emits ultrasound waves, receives their reflections, and converts them into electrical signals. It is used to measure the distance between target objects. The speed of ultrasonic waves surpasses that of audible sound, which is perceptible to the human ear. The ultrasonic sensors contain a transmitter using piezoelectric crystals to produce sound and a receiver that receives the sounds after they have returned (Jost).

Many applications combine proximity sensors and ultrasonic sensors. They also show up in anti-collision safety systems and self-parking automotive technologies. Ultrasonic sensors are also commonly used in robotic obstacle detection systems and manufacturing technologies. In a proximity sensing application, ultrasonic sensors are less affected by smoke, gas and other airborne particles compared to infrared (IR) sensors due to the physical components of an IR sensor being interfered with as opposed to only influenced under certain variables such as heat. To detect, track, and control the level of liquid in closed containers (such as vats in chemical industries), ultrasonic sensors are also employed as level sensors. Most significantly, ultrasound technology has made it possible for the medical field to detect malignancies, create images of interior organs, and monitor the health of unborn children (Jost). Below Figure 2.1 shows the SRF02 Ultrasonic sensor.



Figure 2.1: SRF02 Ultrasonic Sensor (robot\_electronics)

The SRF02 is a tiny PCB ultrasonic rangefinder with a single transducer. It has interfaces for both serial and I2C. The serial interface can be directly linked to any microcontroller's serial ports and operates on a standard TTL level UART format with 9600 baud, 1 start, 2 stops, and no parity bits. SRF02s can be connected in groups of up to 16 on a single I2C or Serial bus. The SRF02 now has two new commands: one for sending an ultrasonic burst without requiring a reception cycle, and another for carrying out a reception cycle in the absence of a previous burst. The SRF02 has a greater minimum range than our other dual transducer rangefinders since it only uses one transducer for transmission and receiving. Like all other rangefinders, the SRF02 has a minimum measurement range of about 20 cm (7.9 inches). It can measure in uS, cm, or inches (Aldahiri, Alrashed, & Hussain, 2021).

### 2.3. Principle of Ultrasonic Sensor

Echolocation is the basis for the operation of ultrasonic sensors. Bats navigate using this idea to avoid clashing with objects. As demonstrated in (Figure 2.2), sound waves are released, and the object's position is determined by measuring the time it takes for the waves to strike and return. Equation (1) provides the object's distance,  $d$ , from the sensor since it assumes that the sound speed in air,  $v_{\text{sound}}$ , is 343 m/s.

$$d = \frac{(t_{\text{roundtrip}} \times v_{\text{sound}})}{2} \quad (2.1)$$

where  $t_{\text{roundtrip}}$  is the whole amount of time it takes for the wave to strike an item and return to the sensor; the one-way trip distance is obtained by dividing the value by two. Accurate findings are obtained from this time-of-flight measurement using sound waves greater than 20 kHz. The accuracy of the ultrasonic sensor, along with the benefits mentioned below, make it a feasible, dependable, and useful choice for human detection.

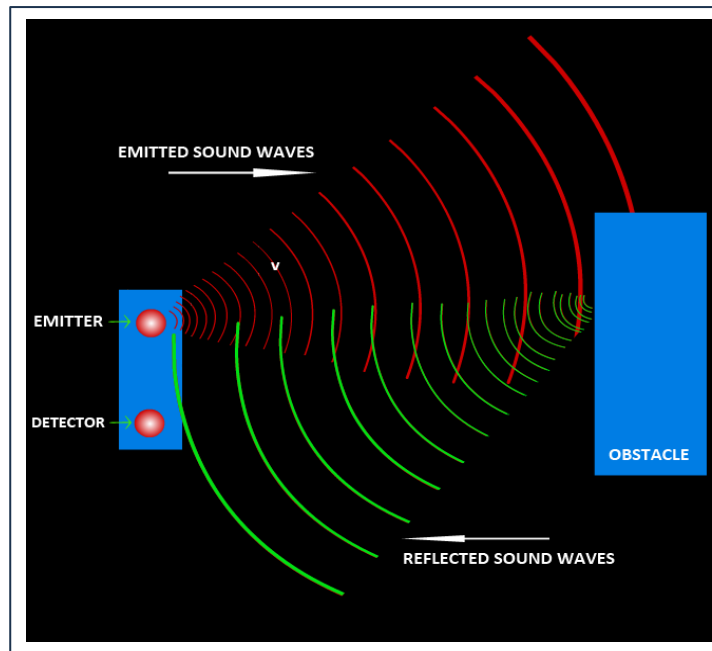


Figure 2.2: Principle working of Ultrasonic Sensor (rolling robot, 2021)

## 2.4. Ultrasonic Data Collection Software

The Ultrasonic Data Collection Software was developed by students(s) in the Master of Information Technology Engineering. (Figure 2.3) illustrates the GUI of the software. The software was set to measure analogue data or raw data from the ultrasonic sensor. Every data set consists of rows representing amplitude values between 0 and 1000 and columns representing frequency values between 34.9 kHz and 44.9 kHz at intervals of 1.08 and 1.09 kHz, altogether yielding 85 columns. (Puthanpurayil,, 2023). In addition to the FFT data, there is exported data headers from the program. Other examples of useful information that may be found in the data headers are things like the model classification result provided by the software, or frequency of sampling (Ilya, Alfred, & Ron, 2006).

The availability of a classification model shows the ability of the program in categorizing or labeling the data collected based on predefined criteria. By the volume of information carried in the header output data, this program comes up as a good solution for ultrasonic data analysis through fusing technical features, such as FFT and categorization, against simple user interfaces through the GUI.

For setting up the GUI software which is used for taking the measurements the following steps are involved:

- When the UDP Client is fired up, and the wireless network credentials for the appropriate sensor are entered, an active "Sensor 1" connection will appear in green at the bottom left.
- The GUI saves the readings in a designated folder. The full path of the folder or directory is '\\GUI\_Vo.23\_2021-11-22\\GUI\_
- -t X Y: Threshold set to obtain the reading has to be specific; X and Y are the different kinds of combinations that are to be used for the threshold values. Using the values of X and Y, this command records the reading to collect data. Additionally, the command "-d 1" allows for the use of demo mode, while the command "-d 0" disables demo mode. Click the "send cmd" button after selecting thresholds, to save the configuration which can be set for different values.
- Measurements: In this block, one set of measurement data corresponds up to 200 measurements per threshold and variation. This mapping demonstrates the total number of readings taken below each threshold within each use case.
- The file name can be altered and modified for each reading by changing it in the blank column. To record the readings, the "ADC" checkbox needs to be selected before pressing the "Start ADC " button. The GUI software automatically stops taking measurements once all of them have been completed, and the file is stored with the assigned name.

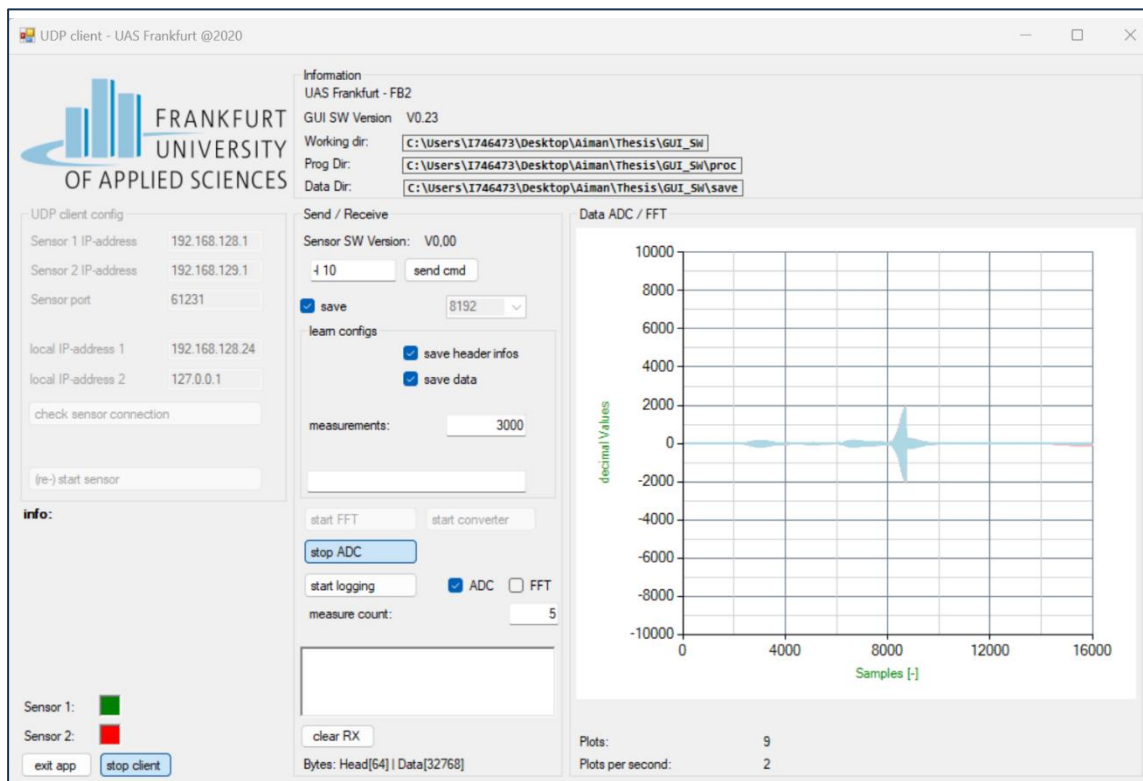


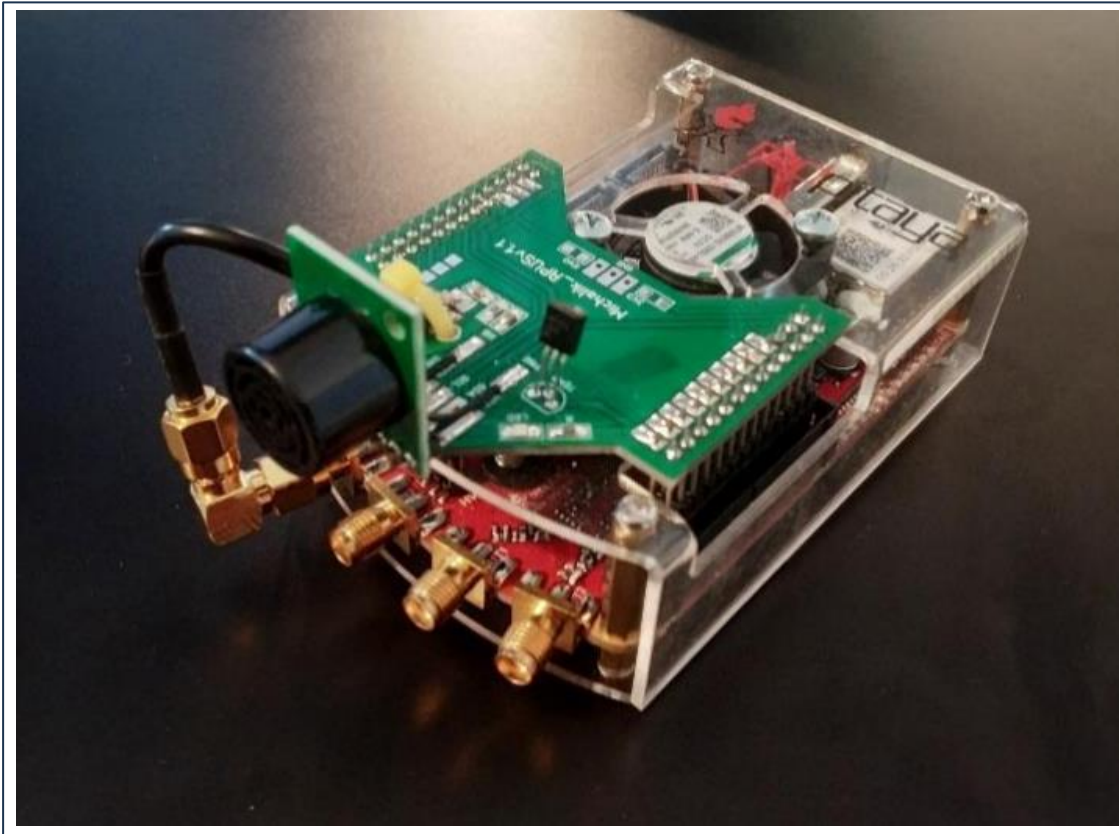
Figure 2.3: GUI for capturing dataset.

## 2.5. Red Pitaya Sensor

Red Pitaya is a popular open-source hardware platform with many applications, as in signal processing or data acquisition and control systems. It is a low-cost analog signal generating and measuring microchip STEM Lab board. Because of its twin fast ADC (Analog-to-Digital Converter) and dual fast DAC (Digital-to-Analog Converter), it is straightforward to acquire two sine waves with arbitrary relative phase shifts along the same channels at once. It is powered by Linux, and it can be controlled or accessible in several different ways such as a web server (running on browsers through your PC/tablet), usb-serial console purposes and SSH protocol (red\_pitaya).

The capabilities of Red Pitaya make it a perfect candidate for integration and experimenting with ultrasonic sensors, especially in projects that require person detection, environmental monitoring, and non-invasive sensing. The Red Pitaya high-speed analog inputs make it possible to acquire the ultrasonic signals even more precisely. They can be sampled at a high rate, a crucial requirement due to the fast reflections that can occur due to the very fast speed of the ultrasonic waves. Additionally, with the FPGA available on the Red Pitaya device, it is possible to program the device to do real-time signal processing applications, such as filtering, which can also enable envelope detection for identifying the echo, required for the accurate measurement of distance and presence. Figure 2.4 shows a Red Pitaya based ultrasonic sensor. Red Pitaya (RP) offers many benefits such as:

- Integration with various programming languages and software environments.
- Well, developed, open libraries and communities, and resources required for accelerated developments. Integration with other sensors and systems for monitoring and control, which proves to be valuable.



*Figure 2.4: Red Pitaya with inbuild ultrasonic sensor.*

## 2.6. Machine Learning Approach

Machine learning (ML) is the field of study that helps computers learn automatically without explicit programming. Notably, machine learning is a subfield of computer science that emerged from the fields of artificial intelligence and pattern recognition. Computational statistics and machine learning are related fields that focus on prediction. Natural language processing, computer vision, pattern recognition, cognitive computing, and knowledge representation are the main areas of interest for machine learning research today. In industrial settings, machine learning techniques are sometimes called predictive modeling (Thomas Rincy & Gupta, 2020).

Depending on the methodology employed throughout the learning process, machine learning can be classified. The study identified four primary categories, namely reinforcement learning, unsupervised learning, semi-supervised learning, and supervised learning (Farhat, Mourali, Jemni, & Ezzedine, 2020).

Furthermore, ML is a more expansive domain that includes Deep Learning (DL). The main objectives of deep learning algorithms are knowledge acquisition and data-driven predictions or judgments. These algorithms take inspiration from the structure and function of the human brain, particularly neural networks. The below diagram shows a general structure of machine learning based predictive model depicting both training and testing phase. A general framework for the machine learning base predictive model is showed in Figure 2.5 below:

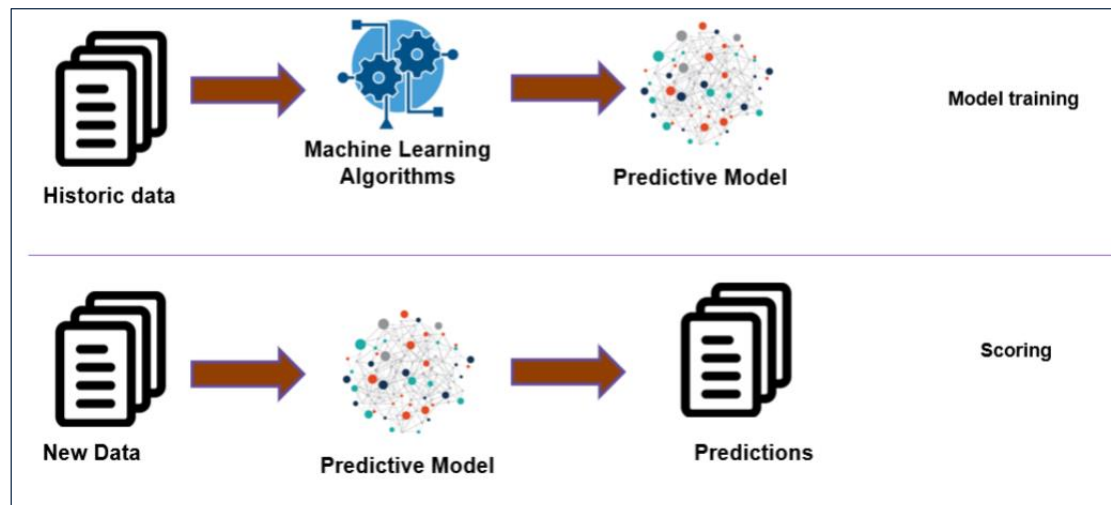


Figure 2.5: ML model with testing and training phases (Aggarwal, 2023)

In Machine Learning (ML), a system can be designed in such a way that as soon as any set of input is fed into it, it begins processing and understanding as an experience, with no help. Primarily, there are two steps involved in building an effective model of machine learning: training and testing. During the training phase, which is a highly research-intensive phase, the system is given labelled or unlabeled inputs. These training inputs are then kept in the feature space by the system for use in upcoming predictions. The system is supplied an unlabeled input at the end of the testing phase, and its task is to anticipate the proper output (Aldahiri, Alrashed, & Hussain, 2021). The major categories of ML are described below:

- Supervised Learning:** The supervised learning model is the most crucial machine learning model. Its primary purpose is to support practical applications. This model is used to forecast results based on two input/output examples and specific sets of given input. Each of the various training datasets involved in supervised training consists of two input goals; one is an input vector, and another is a desired output value referred to as a supervisory signal. Based on these instances of training datasets, ML algorithms are trained with the analyzed instances that provide some inferred classifier function. During the training of algorithms in the case of supervised learning, the goal will be predicting the value of one or more outcomes by making use of different input features (Aldahiri, Alrashed, & Hussain, 2021).
- Unsupervised Learning:** The discovery of unmarked hidden structures in data is one of the uses of unsupervised machine learning. This has been used in several successful applications, although they are frequently challenging to assess. This is a result of inadequate instruction on the application of unsupervised machine learning. Therefore, no error or reward indicators are available for analyzing potential solutions (Aldahiri, Alrashed, & Hussain, 2021). In this thesis, the Multilayer Perceptron which lies under the sub-category of Unsupervised Learning is used.



## 2.7. Multi-Layer Perceptrons (MLP)

The multilayer perceptron algorithm is better known as Feed Forward Network, which is the most common neural network model. It is composed of layers of nodes, with each one examining input before forwarding it to the next layer. Since MLP is a supervised learning algorithm, it uses knowledge learned from labeled data to make predictions on new untouched data. In an MLP, the information comes in via the input layer. The data is passed on to one or more hidden layer processes before finally reaching the output layer. Each layer calls its neurons nodes and connects to neurons in lower layers. Connectivity among these nodes is provided with weights to enhance the accuracy of the model while training.

MLP is a powerful algorithm that may be used for a wide range of tasks, such as regression and classification, and can identify complex patterns in data. It is widely used in speech and picture identification, natural language processing, financial forecasting, and other fields. One potential drawback of MLP is that it might be computationally taxing and require a large amount of data to train well (Nimmada & Basha, 2024).

Since every unit in a layer is connected to every other unit in the layer before it, an MLP has several fully connected layers. Every unit in a fully linked layer has its own set of weights since its parameters are independent of those of the other units in the layer. A multilayer perceptron network with three layers is seen in Figure 2.6.

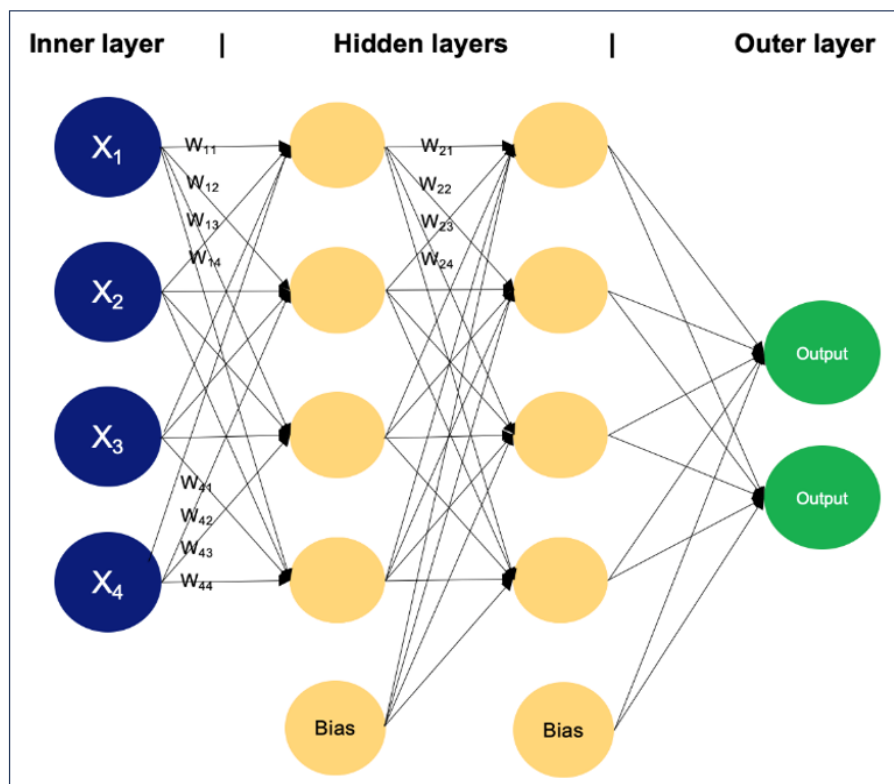


Figure 2.6: MLP with two hidden layers (Jaisawal)

A network of neurons known as perceptron's makes up an MLP. The perceptron is a binary classifier that uses its weighted sum to calculate a single output from several inputs (as well as the constant term "bias," which is independent of any input value).

$$y = \phi(\sum_{i=1}^n w_i x_i + w_0 b) \quad (2.2)$$

Where:  $y$  is the output value,  $\phi$  is activation function,  $w$  is a weight vector,  $x$  is input vector,  $b$  is bias. For time series prediction, a few different MLP designs are available. The output value of a single neuron in an output layer can be interpreted as the time series expected value in the next instant, or a group of neurons can be used to represent the time series values that are predicted in a few following moments. Additionally, the number of neurons in the input layer can vary.

- **Activation Functions**

Each neuron within the hidden layers and also output layer will apply an activation function to a weighted sum of inputs. Some of the most widely used activation functions: sigmoid, tanh, ReLU (Rectified Linear Unit), softmax. There are also non-linear functions applied to the output of each neuron when one forward propagates an input through the layers, and this is what you would be doing anyway if you were training a neural network. This means that the activation function is an output defined by a set of inputs. Below are the most common activation functions with a more detailed explanation. (Kurniawati, Suprpto, & Yuniarno, 2020).

- **Sigmoid Function:** In the Sigmoid function, the range of real numbers comes out to be between 0 and 1. This means any output probability between 0 and 1 can be predicted using this model. Below is how the sigmoid function looks: (Kurniawati, Suprpto, & Yuniarno, 2020)

$$f(x) = \frac{1}{1+e^{-x}} \quad (2.3)$$

- **Rectified Linear Unit (ReLU):** ReLU is currently the most popular activation function. Almost every deep learning or convolutional neural networks use ReLU. In the function, its range value goes in between 0 and infinity, where if  $x$  is less than 0, then the value is 0; and if  $x$  is equal to or above 0, then the value is  $x$ . The ReLU function looks like this: (Kurniawati, Suprpto, & Yuniarno, 2020)

$$f(x) = \max(0, x), \quad (2.4)$$

- **Softmax activation:** A generalization of logistical functions into many classes is the softmax function. As a result, the output can depict how the categories are distributed over numerous classes. This is how the Softmax function looks: (Kurniawati, Suprpto, & Yuniarno, 2020)

$$f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \quad (2.5)$$

## 2.8. Confusion Matrix

Confusion Matrix is a generic process towards testing the performance of a classification-based machine learning model. In tabular form, the confusion



matrix shows how many predictions are correctly and incorrectly done by a classifier or, alternatively, done by a classification model inducted for binary classification problems. Let  $N$  be the total number of target classes, and basically, the confusion matrix is a  $N \times N$  matrix used to determine how well a classification model works. The confusion matrix's total number of target classes can be shown, and by counting the diagonal values for accurate classifications, one can assess the model's accuracy (Xiong, 2012). Figure 2.7 below shows confusion matrix.

|        |               | Predicted                             |                                      |
|--------|---------------|---------------------------------------|--------------------------------------|
|        |               | Negative (N)<br>-                     | Positive (P)<br>+                    |
| Actual | Negative<br>- | True Negatives (TN)                   | False Positives (FP)<br>Type I error |
|        | Positive<br>+ | False Negatives (FN)<br>Type II error | True Positives (TP)                  |

Figure 2.7: Confusion Matrix (NB Share)

Moreover, it can be said that confusion matrix is an essential tool for assessing how well machine learning classification models perform. It has various parameters which are described below in detail. True positives (TP) and true negatives (TN) are listed together with the number of samples that the model properly identified. Moreover, it provides a count of the number of examples that were mistakenly classified as negative (False negative, FN, or Type 2 error) as well as mistakenly classified as positive (False positive, FP, or Type 1 error) respectively.

- **Recall:** The proportion of truly recognized positive predictions comparing to all the existing true positive cases is measured on recall. Through recall, the proportion of real spam emails that the spam filter has correctly identified serves as an important parameter. (Dagang)

$$\text{Recall} = \frac{TP}{(TP+FN)} = \frac{TN}{P} \quad (2.6)$$

- **Specificity:** The classifier's specificity can be defined as the efficiency of the system to evaluate the false (negative) tests, if they are essentially false (Dagang).

$$\text{Specificity} = \frac{TN}{TN+FP} = \frac{TN}{N} \quad (2.7)$$

- **Precision:** The percentage of true positive forecasts among all positive predictions is known as precision. Precision indicates what proportion of emails reported as spam were, in fact, spam in our spam filter example (Dagang)

$$\text{Precision} = \frac{TP}{(FP+TP)} \quad (2.8)$$

- **Accuracy:** Accuracy is the most fundamental measure for assessing a classification models. It shows the proportion of accurate predictions your model has made. Utilizing a confusion matrix, the accuracy can be determined using the following formula (Dagang).

$$\text{Accuracy Score} = \frac{(TP+TN)}{(TP+FN+TN+FP)} \quad (2.9)$$

- **F1 Score:** F1 score is a measure of performance that summarizes the recall and precision of a model. In other terms, the F1 score is the harmonic average of the two measures. The reason for combining recall and precision into a single statistic was to make the performance evaluation of a model without explicitly stating the two (Dagang).

$$\text{F1 score} = = \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (2.10)$$

## 2.9. Kalman Filter

The Kalman filter is an algorithm used to estimate the underlying states of a system based on noisy observations. It operates recursively, updating the state estimate as new observations become available. The Kalman filter algorithm consists of two steps: the prediction stage and the update stage (Turing)

- **Prediction Stage:** The initial phase, known as the prediction stage, involves forecasting the state components by utilizing data available up to time  $t-1$ . Four equations make up this step. The first step predicts the state components using the time series properties. The second step involves forecasting the covariance matrix of the state components. Third, is the prediction error of the time series of interest. Lastly, we have the variance of the prediction error (Hubbard)
- **Update Stage:** The updating step, which comes in second, produces predictions using all available data as of time  $t$ . It is composed of three formulas. Based on the complete set of data, the first step predicts the state components, and the last step is the updating of the covariance matrix of the state components.

Kalman Filter (KF) is a method that offers a solution in the following areas:

- **Object Tracking** – If you know where on the coordinate plane the object you are interested in is situated, you are one step away from getting the object's motion and velocity. (Franklin, 2020)

- **Guidance, Navigation, and Control** – The IMU, as the Inertial Measurement Unit is often called, provides appropriate data for such needs as finding the velocities of the object and its acceleration. (Franklin, 2020)
- **Body Weight Estimate on Digital Scale** – This process covers the use of the pressure assessed on applied on a flat surface to manufacture a consistency of certain object weight. (Franklin, 2020)

### 2.9.1. Application in Noise Suppression

One of the most useful applications for Kalman filters is noise suppression in ultrasonic sensors. The data generated by ultrasonic sensors is quite noisy for several reasons, i.e., environmental artifacts also multipath reflections and sensor constraints. This noise degrades performance of ML model that depends on data for accurate classification. The Kalman Filter helps in smoothing out the noise and gives a clean signal which is very close to what would happen.

To solve these difficulties, this thesis aims to provides the detailed description of the Kalman filter model that not only filters out noise but also estimates sound pressure from ultrasonic sensor data. The Kalman filter can quickly update and make a representational estimate of the sound pressure in the environment by discriminating between noise and true imagery. This dual functionality strengthens the reliability of data which drives person detection, since both noise-reduced signal and estimated sound pressure benefit a more robust detection system.

### 2.9.2. Sound Pressure Estimation

While sound pressure depends on many factors, among which are the distance between the sensor and reflective surface, the velocity of the sound wave, and properties of the medium surrounding the receiver-air. In this project, the smoothed signal data obtained from the Kalman filter is used for computing the sound pressure, coming up with precise estimates of the state of this system-pressure, velocity, and acceleration.

## 2.10. Thresholding Methods

The major thresholding methods which are utilized in this thesis are described briefly below:

1. **Niblack's Thresholding Algorithm:** The local mean and local standard deviation are the foundations of Niblack's algorithm, a local thresholding technique. The following formula determines the threshold (Ilya, Alfred, & Ron, 2006)

$$T(x, y) = m(x, y) + k \cdot s(x, y), \quad (2.11)$$

where  $s(x, y)$  and  $m(x, y)$  are the standard deviation and local area values, respectively. The community should be both large enough to block out noise and tiny enough to preserve local characteristics. The amount of the total print object boundary that is included in the given object can be changed by varying the value

of  $k$ . The enhanced variant of the Niblack algorithm is as below:

$$T(x, y) = m(x, y) \cdot \left[ 1 + k \cdot \left( 1 - \frac{S(x, y)}{R} \right) \right], \quad (2.12)$$

where the empirical constants  $R$  and  $k$  are used. The parameters  $k$  and  $R$  are used by the modified Niblack method to lessen its susceptibility to noise.

Niblack local thresholding is important image processing algorithm for the binarization process. The method designed by Niblack is capable of handling adjustment to local contrast and illumination changes in an image by setting the threshold of the pixel depending on its neighborhood. It clearly indicates that it offers local adaptability in binarization but is different in approach from the global threshold methods where the entire image has only one threshold imposed upon it (Ilya, Alfred, & Ron, 2006)

1. **Yanowitz-Bruckstein's Adaptive Thresholding:** (S.D. & A.M., 1989) presented a novel technique for segmenting non-uniform images that relies on potential surface interpolation and a novel adaptive thresholding method. Ultimately, the following equation was used to get the segmentation result:

$$P_n(x, y) = P_{n-1}(x, y) + \frac{\beta \cdot R(x, y)}{4} \quad (2.13)$$

Because of the differences in contrast between their surfaces, this strategy offers an improvement over a fixed threshold (Mustafa, Khairunizam, Ibrahim, A. B., & Razlan, 2018)

Nevertheless, Adaptive thresholding by Yanowitz and Bruckstein is a more sophisticated method, taking into consideration both spatial and intensity information from the image. Contrasting with the global thresholding methods that use one value of the threshold for the whole image, locally, the Yanowitz-Bruckstein method adjusts the threshold value based on the distribution of pixel intensities and the local structure of the image. The first step of the method identifies initial seed points in the image, used to construct an initial threshold surface. Seed points are handpicked around the high-contrast regions with known background and foreground regions. Algorithm then refines that surface iteratively by threshold propagation through the image (Ilya, Alfred, & Ron, 2006)

## 2.11. Signal Envelope Process

In signal processing, the envelope of a signal is referred to the graphical implementation showcasing the upper and lower bounds of oscillatory signals. This is especially useful when one deals with high-frequency signals, as provided by ultrasonic sensors. The envelope adds another layer of information by smoothing the oscillations and enhancing the trend or amplitude variations in the signal as it varies in time.

The envelope helps trace the general trend of the sound pressure data, by which it is much easier to tell the peaks corresponding to occupancy from surrounding noise, for example, the peaks due to Niblack and Yanowitz-Bruckstein. This becomes particularly useful when one is dealing with filtered data where some noise has been cut off but where peaks need to be emphasized.

The envelopes of a signal as in Figure 2.8 the boundary within which the signal is contained, as an imaginary curve. Envelopes contain some information of signals, though it is an imaginary curve, for example, demodulating amplitude modulated (AM) signals by them. However, the envelope used in technical problems is not conceptually as clear-cut as the geometrical concept of the envelope of a family of curves (Vasavi, Nikhita, & Sai Krishna, 2024)

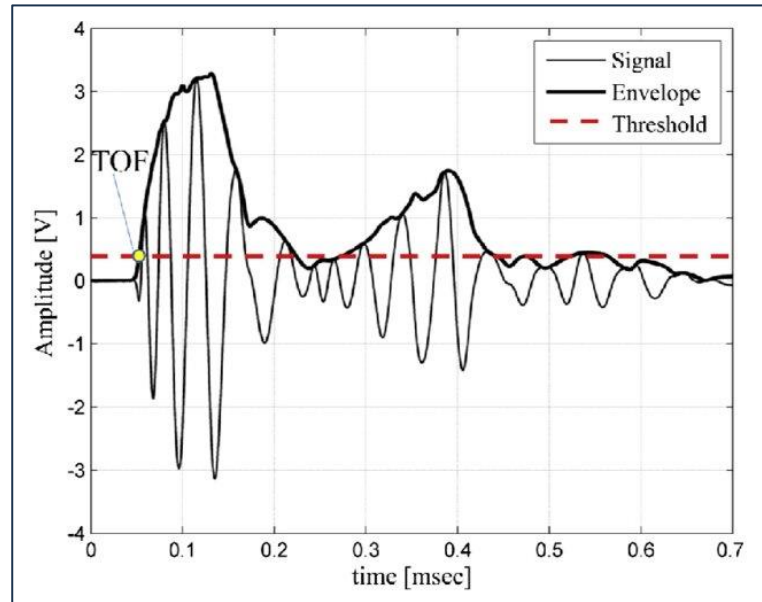


Figure 2.8: Signal and its corresponding envelope (Dehghan-Niri, Ehsan, Alireza, & Salvatore, 2013)

### 2.11.1 Application of Signal Envelope

1. **Peak Prominence Detection:** The envelope emphasizes, by a visual effect, the upper bounds of the signal. In this way, the envelope helps to detect the most prominent peaks, which correspond to the significant events-that a person sat on or left the chair.
2. **Graphical Representation of the Amplitude Variation:** The envelope in that sense gives a clear insight into the amplitude variation of the signal with time and thus plays an important role in the differentiation of the states whether occupied or non-occupied.
3. **Feature Extraction Enhancement:** Carrying this out on the envelope, as opposed to on the raw signal, greatly simplifies the extraction of such features, which could include timing and magnitude of key peaks. The extracted features are then used for classification by the MLP classifier.

## 2.12. Graphical User Interface (GUI)

GUI, a graphical user interface, is simply an interactive interface that allows users a visual bridge in interacting with software- based or electronic machines. GUIs, short for Graphical User Interfaces, are aimed at enabling users to work with the process without necessarily being well conversant with hardcore programming or

command line interface interaction. With these tools, the user can make the task less technical and more user-friendly to the different categories of users including non-technical professionals. (Hauptmann, Hoppe, & Puettmer, 2001)

Tkinter provides extensive possibilities for the easy construction of flexible GUIs by Python developers. Known for its simplicity and ease of use, Tkinter has become a usual choice to develop interactive applications-ranging from simple tools up to complex desktop applications. Its rich set of widgets grants Tkinter an event-driven architecture and perfect integration with other libraries, turning it easily into the ideal solution. It will provide a framework for crafting visually appealing and responsive interfaces (Hauptmann, Hoppe, & Puettmer, 2001)

## Chapter 3

# Literature Review

### 3.1. Occupancy Detection

Detection of occupancy, in recent years, has a long way to go, but its history can more easily be seen in optimization of energy, augmentation of safety and security, and personal comfort. The historical journey into this area has its major turning points and key changes that brought us today to the development of occupancy detection systems. Initially, the detection of occupancy was limited to simple techniques only, which were based either on manual count or on/off switch. The initial techniques were surely cumbersome and prone to false counts; hence, this was feasible for large-scale applications. Nevertheless, with the development in technologies, better solutions started to appear for occupancy-based control systems (Kamthe, Jiang, Dudys, & Cerpa, 2009)

Occupancy information in buildings may bring significant impacts on both energy consumption and indoor environment quality (Yang, 2017). Many studies indicated that about 10%–40% of energy consumption in buildings can be saved if occupancy information is available. For instance, energy can be saved by switching off the lighting system during non-occupied time; adjusting the opening rate of air damper or air conditioner based on the occupancy number information (Kailai Sun & Zhao, 2020). Occupancy monitoring can be categorized mainly into group-based and individual monitoring. The former refers to estimation of the aggregated occupancy in space, while the latter refers to tracking each occupant's position and identification of each occupant. Certain occupancy estimation systems require equipment such as a mobile phone or radio frequency identification tag that ought to be provided by the end users.

About 40% of the world's energy is consumed by buildings, and they emit nearly one-third of the total greenhouse gases worldwide. They are also responsible for using about 60% of the electricity in the world. In fact, over the last decade, quite strict building regulations have resulted in the quality of the thermal characteristics of many building envelopes being significantly improved. On the other hand, similar consideration has not yet been paid to the number and activities of occupants in a building, which increasingly play an important role in energy consumption and optimization processes, besides indoor air quality (Babu, Kumar, & Kumar, 2006).

### 3.2. Ultrasonic Sensor Classification

Specific transducers and a specialized electrical system for signal acquisition and parameter extraction are needed to detect ultrasonic wave's properties. Many ultrasonic systems were previously only possible as scientific instruments, but because to advancements in microelectronics, they are now affordable, small gadgets (Patkar & Tasgaonkar, 2016)

A sound signal must be reflected in order to be measured for distance. A longitudinal sound wave colliding with a flat surface is this sound signal. As long as the reflective surface's dimensions are greater than the sound's wavelength, sound is reflected (Michael & Myeongsu, 2018)

As the ultrasonic waves travel farther across the medium, their power decreases rapidly. The device must rely on the target to reflect the pulse back to itself in order to determine distance. It is desirable for the target surface to be perpendicular to the pulses' direction of transmission (Michael & Myeongsu, 2018)

### 3.3. Kalman Filter in Occupancy Detection Systems

Kalman Filter (KF) based target tracking in Wireless Sensor Network (WSN) is related to recent research (Nabae, Pooyafard, & Olfat, 2008), (Vaidehi, et al., 2010). which states that the person can also be detected by more than one sensor or may not be sensed even by a single sensor. Hence, logic is required to predict the missing events and future position of a person based on the past dynamics.

The Kalman filter further filters the measurement noise and, based on a system model, predicts the next position of the person. For a uniform sampling period, the KF uses the current position of the person to predict his next position. The deployment of sensors may be either uniform or random (Mittal, Prasad, Saurabh, Xue Fan, & Hyunchul Shin, 2012). Recent improvements in object detection have spurred interest in algorithms that would combine tracking and detection. These indeed lead to combinations of object detectors and Kalman filtering, yielding algorithms more suitable for time-critical, online applications (Wang & Zhang, 2023).

### 3.4. Machine Learning

The process by which a computer learns to do tasks given example data is called machine learning. We know that as we give a machine additional experience (E) with a certain job (T), its performance (P) improves (Yang, 2017) Machine learning's advantage over conventional expert systems is its capacity for learning. In this respect, a model is said to be underfitting if it is unable to acquire the knowledge carried by past data, and overfitting if it is able to learn the noise information in historical data. (Farhat, Mourali, Jemni, & Ezzedine, 2020)

It is the field that studies different computer algorithms and makes gradual improvements with experience. Machine learning can be divided into four categories: reinforcement learning, semi-supervised learning, unsupervised learning, and supervised learning (Thomas Rincy & Gupta, 2020)



A machine learning task known as "supervised learning" makes assumptions based on the labeled training data. Unsupervised learning uses unlabeled data, or more specifically, unlabeled data. Combining labeled and unlabeled data is known as semi-supervised learning. The software agent uses information gathered from interactions with the environment to determine how best to maximize rewards in reinforcement learning (Otsu, 1979).

### 3.5. Feature Extraction and Thresholding Methods

Over the past 20 years, numerous binarization methods have been put out. Global thresholding and local thresholding are the two broad categories into which these methods can be divided. Single intensity threshold value is used by global thresholding techniques. To categorize picture pixels into foreground (text) or background (non-text) pixels, this value is computed using a few heuristics or statistics of global image properties (Khurshid, Siddiqi, Faure, & Vincent, 2009). The primary problem with global approaches is that they perform poorly on low-quality document images because they are unable to adjust to irregular illumination and noise (Niblack, 1986).

Niblack's approach slides a rectangle window across the gray level image to determine a pixel-wise threshold. The threshold surface, which is established by interpolating the picture gray levels at locations where the gradient is high and suggests likely object edges, is the basis for the image segmentation technique that Yanowitz and Bruckstein introduced. Because this method relies entirely on the edge of the image, there are bogus objects when the noise is high and objects that are lost when the edges of the objects are too faint (S.D. & A.M., 1989).

# Chapter 4

## Methodology

### 4.1. Objective

The main objective of this thesis is to build a robust and optimize a machine learning based system for person detection by employing data from the ultrasonic sensors available for an office environment. The primary object is to use a Kalman filter to improve the ML model that can accurately classify ultrasonic reflections from occupied and non-occupied office chairs. It employs the Kalman filter for noise suppression and sound pressure estimation; hence, the capability of the classifier in differentiating between an occupied and non-occupied office chair will be boosted significantly.

### 4.2. General Architecture of the system

The Figure 4.1 represents the generic architecture of the person detection system. The system will have a set of components, namely, an ultrasonic sensor that collects the data, a Kalman filter to reduce noise and estimate sound pressure, and an MLP classifier that will classify the office chairs as occupied and non-occupied.

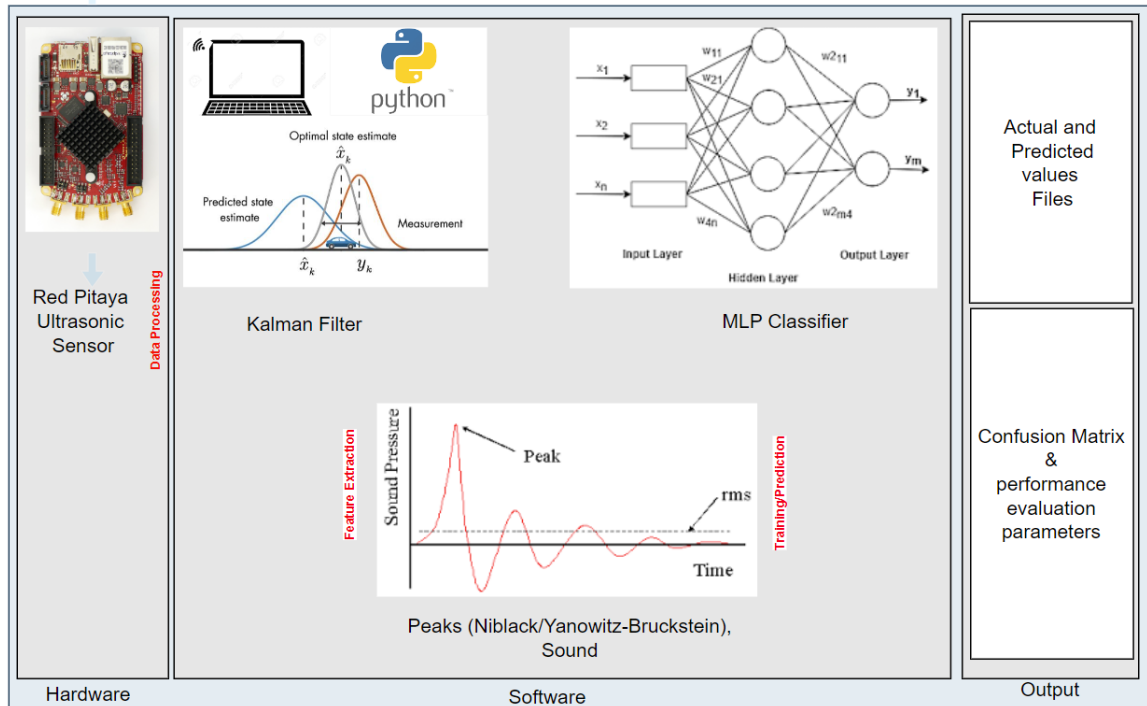


Figure 4.1: General Architecture of the System

The block diagram above shows the basic architecture with various components, each of which serves as a building block in the end-to-end workflow and their smooth compatibility to result high level of person detection accuracy.

### 4.2.1. System Components

#### 1. Hardware Requirements

The hardware requirements define physical components required for the person detection system, sensors, data acquisition platforms, and computation resources. The system depends on ultrasonic sensors as an avenue of data intake, real-time data acquisition with the Red Pitaya board, and a computer system for processing data and machine learning in general.

##### I. Ultrasonic sensor

Ultrasonic sensors are the most important hardware one will need in determining the occupation of a chair. They work by sending high-frequency sound waves above 20 KHz and calculate the time of flight from sending the sound wave to receiving the wave reflected off objects, such as chairs or people. Variation in the sensor data carries some important information about the distance between the sensor and the reflecting surface, which changes with the occupancy of the chair.

##### II. Red Pitaya

The Red Pitaya (RP) is multipurpose in it interfaces with ultrasonic sensors, performing very fast data acquisition and processing. It is a Red Pitaya device that performs multiple tasks of acquiring, processing, and analyzing. Conclusively, the Red Pitaya makes it possible to convert the raw data supplied from the ultrasonic sensor to FFT or ADC data according to the features of the sensor data and prepare it for further analysis. Because it offers multifaceted capabilities of signal processing, it efficiently takes care of data, thus prepping for further steps. The Figure 4.2 shows Red Pitaya Software GUI.

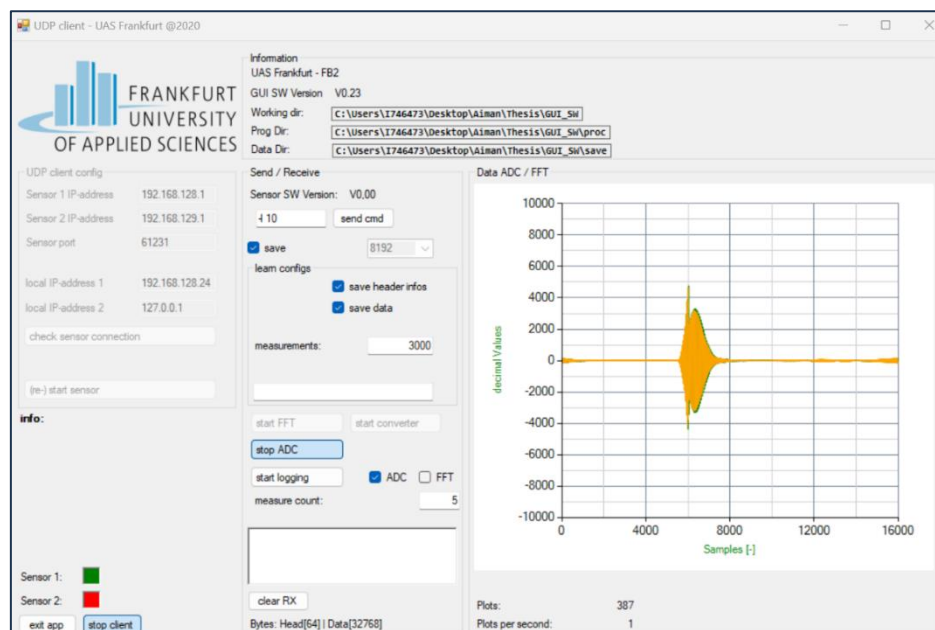


Figure 4.2: Red Pitaya GUI for measurement.

## **2. Software Requirements:**

This section gives details about the software requirements including information regarding tools, programming languages, and libraries to realize the design, implementation, and testing of the ultrasonic person detection system. The foundation elements that make up the entire system are signal processing, machine learning, and data visualization, which all demand high-performance performance with specific software and libraries.

### **I. Programming Language and Libraries**

Python is used as a main implementation language for the algorithms of signal processing-such as the Kalman filter-feature extraction methods, like Niblack and Yanowitz-Bruckstein-and the classifier Multi-Layer Perceptron (MLP). Python has been used because of its simplicity, flexibility, and huge number of libraries that exist to support data processing, machine learning, and visualization. Different python libraries such as NumPy, Pandas, Scikit-learn, Matplotlib, Tkinter were used in the experimental process to implement different components of the system.

### **II. Kalman Filter Model and Feature Extraction Methods**

The Kalman filter is a state estimation algorithm that filters out the noise in the data provided by ultrasonic sensors, making an estimation for some important state variables such as pressure, velocity, and acceleration. Furthermore, the Kalman filter model estimates the sound pressure taking into consideration temperature and the distance of the sensor with respect to the reflective surface. The code for Kalman filtering was integrated into the general preprocessing chain of data to clean and properly prepare sensor data toward feature extraction and classification.

Once noise-free state measurements have been obtained using a Kalman filter, subsequently extracted features are calculated based on the clean and accurate measurements. A combination of Niblack and Yanowitz-Bruckstein peak detection methods extracts important features from the data processed by the Kalman filter. These are the features that are used as input for classifier to classify any occupied or non-occupied space.

Niblack thresholding technique uses the analysis of the local mean and standard deviation within a window to detect the peaks within the ultrasonic signal. Numerical operations are accelerated using NumPy. This step is especially helpful in the detection of transitions from an occupied to a non-occupied state and vice-versa. On the other hand, the Yanowitz-Bruckstein is a window-based peak detection method that selects the most relevant events from the signal. Both are integrated into the feature extraction pipeline that produces the input features for the MLP classifier.

### **III. MLP Classifier**

Multilayer Perceptron (MLP) is used as a classifier in this thesis as a classifier to predict the occupied and non-occupied office chairs. It is a feedforward neural network model which processes the features extracted (peaks and sound pressure estimates) from the data and classifies this into a state of chair either

occupied or non-occupied. The MLP is initialized with pre-trained parameters in a labeled dataset and fine-tuned for the target domain specific high-accuracy performance that can be obtained using real-world training data.

### 3. Output Comparative Analysis

The work is developed in Python and its related libraries so that there is smooth integration among the components involved: data acquisition, data preprocessing, and classification. Reporting the classifier's decision is the last component's responsibility. Using the prediction from the MLP classifier, the system gives a binary result, with 1 denoting non-occupied and 2 occupied. The Results-accuracy, confusion matrix, and other performance metrics-are visualized to make a comparative analysis with and without Kalman filter. Uses for this output in real-time include security monitoring, workplace environment management, and energy-saving devices (such shutting off lights while a chair is not in use).

## 4.3. Physical Experimental Work Setup

To be sure that the data being provided by sensors was consistent, the experimentation activity was done under controlled lab conditions. The sensor of Red Pitaya chip was placed at a chair level of approximately 1.5 m to collect accurate information. The focal point of the arrangement was a chair that was positioned immediately underneath the sensor. The Red Pitaya is powered by connecting it to the power supply. Data collection process was aided by the integrated sensor system, which also has a graphical user interface (GUI).

Two scenarios have been used to collect the data: one involves a human sitting on the chair which is identified as the "occupant," and the other involves an empty chair or any other object that is identified as the "non-occupant." The scenarios tested, considered different settings, participant demographics, and several similar situations. The experiment followed these parameters strictly in order to minimize extra influences that could compromise the validity and reliability of data coming from sensors. This helps to understand the range and level of variability of the test conditions within which the data was obtained. Due to the fact that every condition was documented with much care, the occupancy detection model was able to generalize well in different situations and therefore made the model robust and dependable for a smart office environment.

## 4.4. Dataset Acquisition

In this phase, the raw dataset is collected using the Red Pitaya sensor for various use cases. The dataset is further used for preprocessing using Kalman filter and after the analysis, a confusion matrix is generated, and conclusions are drawn. A total of four datasets and their details are described below:

### A. Case 1: Non-Occupied (Empty chair):

This case contains all the datasets which were taken while keeping the office chair empty (non-occupied chair)

|           |                                                               |                                |                            |
|-----------|---------------------------------------------------------------|--------------------------------|----------------------------|
| Dataset 1 | Non-Occupied chair (Empty)                                    | Window Open/<br>Windows Closed | Lights<br>On/Lights<br>Off |
| Dataset 2 | Non-Occupied chair (Chair with<br>cardboard boxes kept on it) | Window Open/<br>Windows Closed | Lights<br>On/Lights<br>Off |

### i. Dataset 1

To record this dataset, an Empty chair was kept just below the sensor at a certain height and considering the other environmental conditions such as the lights of the room was kept switched on while performing this experiment. The window adjacent to the office chair was also kept open as well as closed in another scenario to take into consideration the external noise. A total of 5000 readings were recorded for each case for further processing. In each use case only one parameter is changed at a time and readings are taken to analyze the impact that parameter on the sensor's classification efficiency. Below shows the folder structure of the captured dataset for this case as in Figure 4.3.

| <div> <div> <div></div> <div></div> <div></div> <div></div> </div> <div> <div>Sort</div> <div>View</div> <div></div> </div> </div> |                  |          |            |
|------------------------------------------------------------------------------------------------------------------------------------|------------------|----------|------------|
| Name                                                                                                                               | Date modified    | Type     | Size       |
| <div> <div></div> <div>Empty_Chair.txt</div> </div>                                                                                | 23/07/2024 14:23 | TXT File | 256.132 KB |

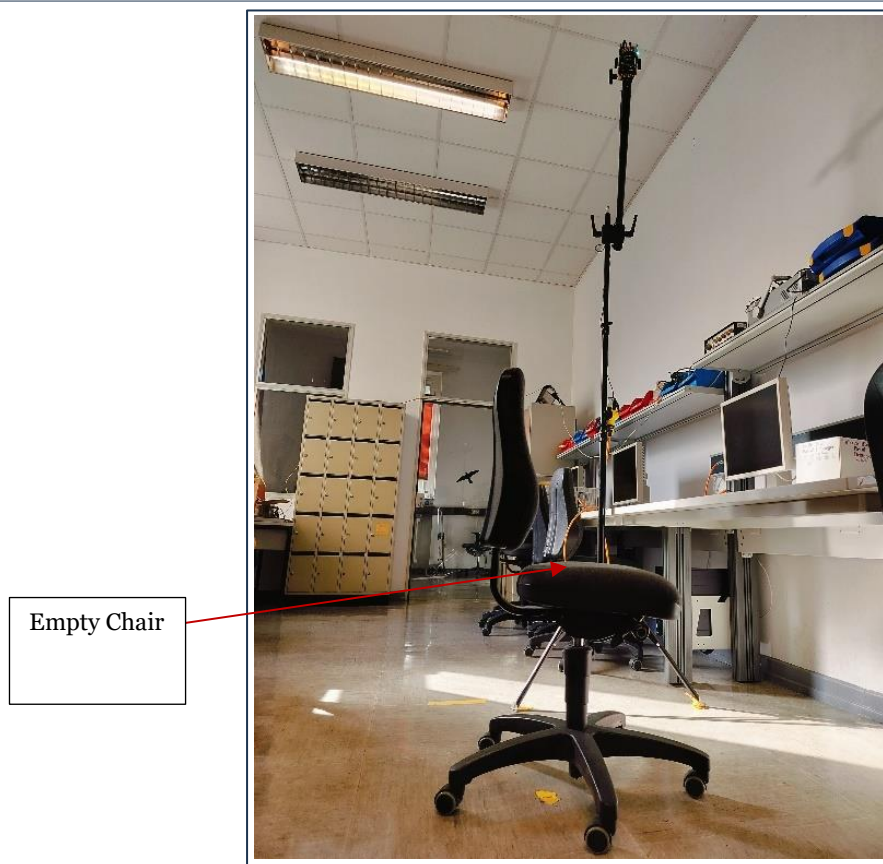


Figure 4.3: Experimental setup with empty office chair

## ii. Dataset 2

In this case, the reading was recorded by keeping the cardboard boxes on an empty chair as in Figure 4.4. In this case as well, the environmental conditions such as lights and windows were set differently to take different scenarios.

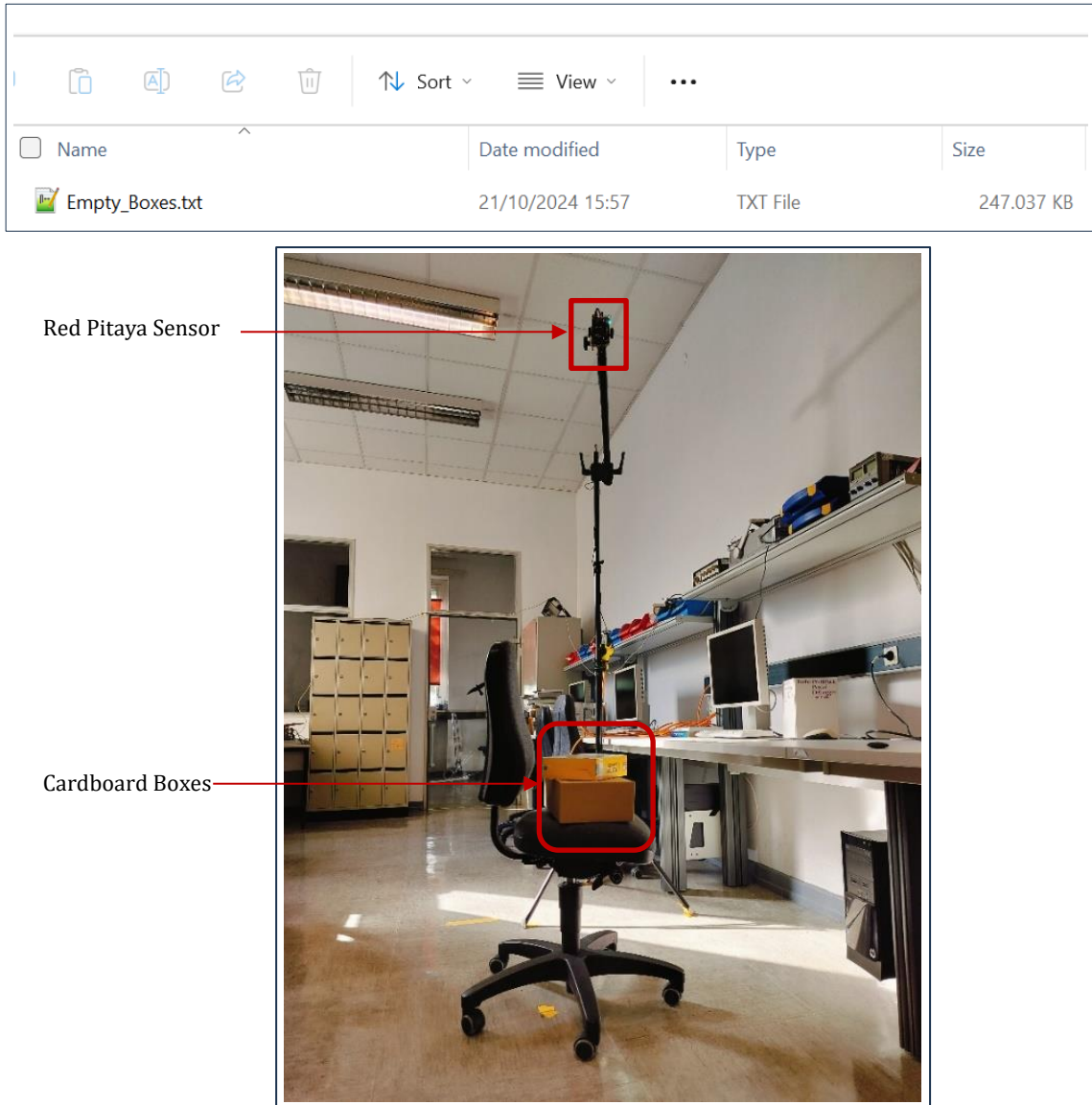


Figure 4.4: Experimental setup with a cardboard box kept on an office chair.

### B. Case 2: Occupied (Human)


This case contains all the datasets which were taken when the office chair was occupied (human sitting on the chair) with different environmental conditions (occupied chair)

|           |                                                                                                                |                                |                         |
|-----------|----------------------------------------------------------------------------------------------------------------|--------------------------------|-------------------------|
| Dataset 1 | Occupied chair (Human sitting idle on chair)                                                                   | Window Open/<br>Windows Closed | Lights<br>On/Lights Off |
| Dataset 2 | Occupied chair (Human sitting on a chair), Moving head/shaking hands, Different obstacles kept on office table | Window Open/<br>Windows Closed | Lights<br>On/Lights Off |



i. Dataset 1

In this scenario, a human is sitting on a chair placed directly underneath the Red Pitaya sensor. The window is kept open, and the lights are kept switched on for the environmental conditions as in Figure 4.5. A total of 5000 readings are recorded for this dataset condition.

| <div><div><div><div></div></div><div><div></div></div><div><div></div></div><div><div></div></div></div><div><div>Sort</div><div>View</div><div></div></div></div> |                  |          |            |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|----------|------------|
| <input type="checkbox"/> Name                                                                                                                                      | Date modified    | Type     | Size       |
|  Human_idle.txt                                                                   | 29/08/2024 11:08 | TXT File | 256.267 KB |

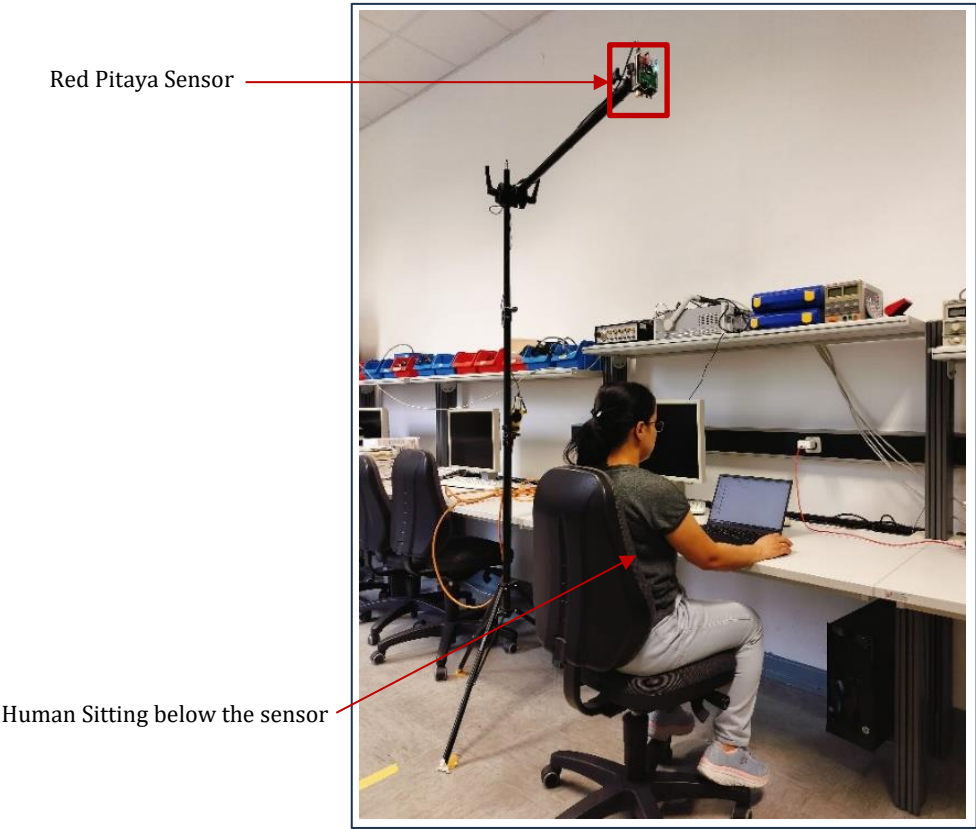



Figure 4.5: Experimental setup with a person sitting idle on an office chair.

ii. Dataset 2

This case comprises of a human sitting on an office chair under the red pitaya sensor which is placed at a certain height. In this case the human sitting under the chair is shaking hands and moving head at different positions. Also, several obstacles were kept on the office chair like the bunch of books, coffee mug, laptop, office bag etc., Here also the windows were kept open/closed and lights were kept switched on/off consecutively to perform the experiment as in Figure 4.6.

| <div><div><div><div></div></div><div><div></div></div><div><div></div></div><div><div></div></div></div><div><div>Sort</div><div>View</div><div></div></div></div> |                  |          |            |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|----------|------------|
| <input type="checkbox"/> Name                                                                                                                                      | Date modified    | Type     | Size       |
|  Human_moving.txt                                                               | 28/08/2024 13:16 | TXT File | 263.357 KB |



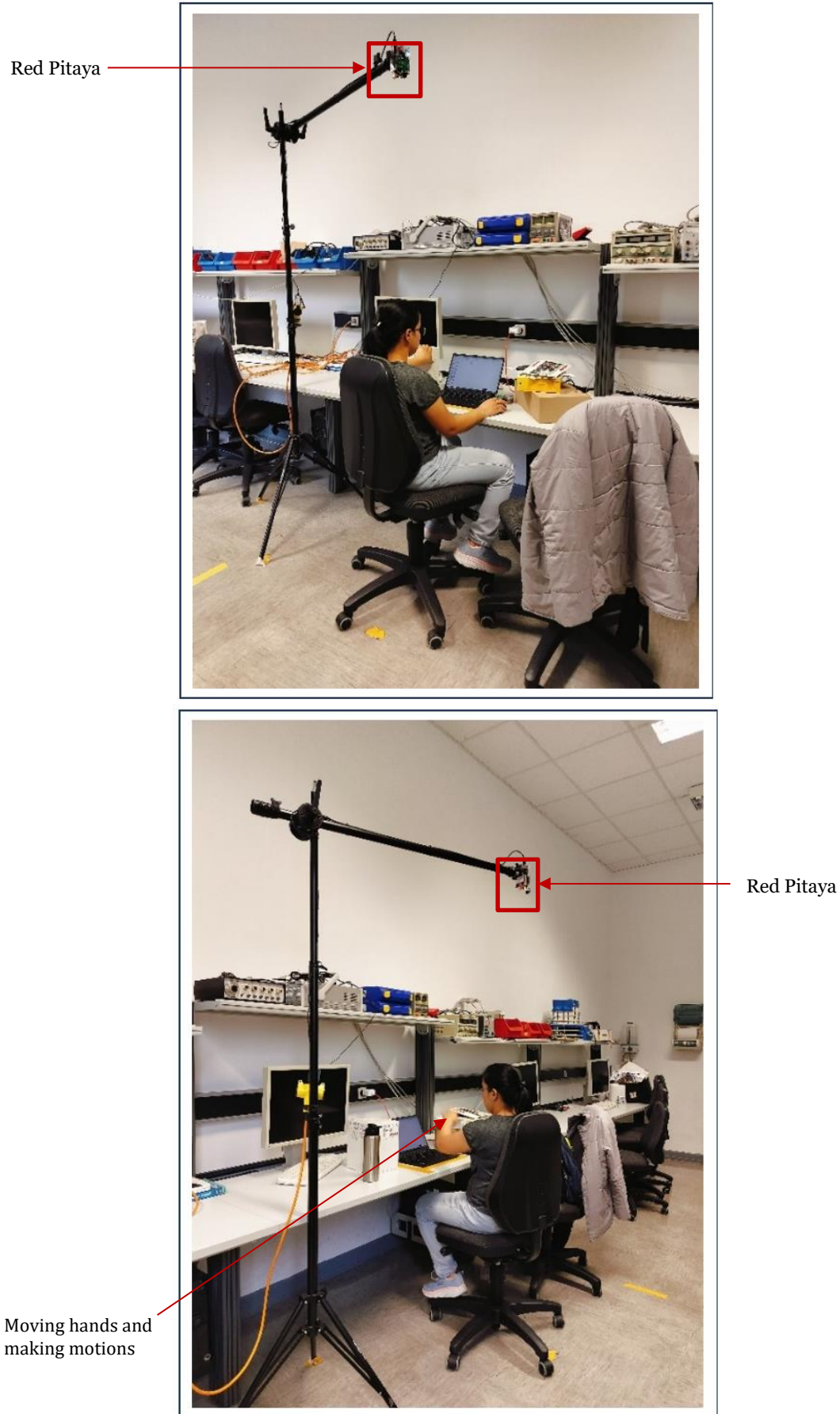


Figure 4.6: Experimental setup with a person sitting on an office chair, continuously moving hands and multiple obstacles kept in nearby vicinity.

## 4.5. Software Setup

Once the raw dataset is obtained, then comes the process of using it for data processing and further other processes. To start with those process, the first step is to step up the software environment in the local machine. For all the codes in the experimentation, python is used as a programming language.

### 4.5.1. Setting up a Virtual Environment

A virtual environment was created to segregate the project from the globally installed version of Python; this may allow compatibility on different machines without any possible conflict between the different versions of packages. The custom environment was created by using Python's built-in **venv** module. The virtual environment makes it easy for us to use and install only the essential libraries which are actually needed for the project and thus, all system libraries are no longer needed to be installed. Below steps were followed to create the virtual environment:

- a) The environment was created using the below command which created an isolated directory for Python and its dependencies.

```
python -m venv <environment_name>
```

- b) To ensure the environment was active during development, it was activated via below command (For windows).

```
.\<environment_name>\Scripts\activate
```

- c) All the python libraries required for specific programs which is discussed in detail in the upcoming topics below, were installed sequentially:

```
pip install pandas
```

```
pip install numpy
```

```
pip install matplotlib
```

- d) The requirements.txt file was generated to manage the dependencies and allow the replication of the environment. Using the below command, this file was created automatically.

```
pip freeze > requirements.txt
```

### 4.5.2. Removal of Headers from the Raw dataset

The data processing is done after all the categories of dataset is collected using ultrasonic based Red Pitaya Sensor served as the primary hardware component. The two categories "Occupied Seat" and "Unoccupied Seat," offer each distinct feature of detected events, and this arrangement includes a GUI interface that makes it easier to gather ADC data for those categories. Depending on the placement of the sensor, a wireless connection is made to one of the sensors with the help of the GUI software. For configuring the GUI measurement, the UDP Client is launched and connection to the sensor is established indicating the green signal. The linked sensor's IP address is 192.168.129.1. and port number is 61231. The sample of a raw dataset obtained from the UDP client is shown below Figure 4.7. The measurement's setting is recorded in the first 16 columns of data in each data file, which are called headers. When the server transmits ADC data, the client receives about 32832 bytes of data. The header

bytes are the first four bytes in the stream that is received, and the data length is the next four bytes. Depending on the 64-byte size of the header as established by the decoding process, the length of the contents is calculated as (32832 - 64), producing 32768 bytes.

|   | A  | B     | C | D | E   | F | G       | H  | I | J    | K    | L | M | N    | O | P |
|---|----|-------|---|---|-----|---|---------|----|---|------|------|---|---|------|---|---|
| 1 | 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 2839 | 1,51 | 0 | 0 | V0,2 | 0 | 0 |
| 2 | 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 2864 | 1,51 | 0 | 0 | V0,2 | 0 | 0 |
| 3 | 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 2839 | 1,51 | 0 | 0 | V0,2 | 0 | 0 |
| 4 | 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 2864 | 1,51 | 0 | 0 | V0,2 | 0 | 0 |
| 5 | 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 2864 | 1,51 | 0 | 0 | V0,2 | 0 | 0 |
| 6 | 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 2839 | 1,51 | 0 | 0 | V0,2 | 0 | 0 |
| 7 | 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 2839 | 1,51 | 0 | 0 | V0,2 | 0 | 0 |
| 8 | 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 2839 | 1,51 | 0 | 0 | V0,2 | 0 | 0 |
| 9 | 64 | 32768 | 1 | 1 | 512 | 0 | 1953125 | 12 | 0 | 2839 | 1,51 | 0 | 0 | V0,2 | 0 | 0 |

Figure 4.7: Raw dataset from the UDP Client

Table 4.1: Detailed Headers Description

| No. | Headers                                                                  | Length (byte) | Value             | Unit                             |
|-----|--------------------------------------------------------------------------|---------------|-------------------|----------------------------------|
| 1   | Header length (count for fields)                                         | 4             | 64                | -                                |
| 2   | Data Length                                                              | 4             | 32768             | ADC_DATA: 32768<br>FFT_DATA: 170 |
| 3   | Class detected                                                           | 4             | 1 or 2            | 1: Object<br>2: Human            |
| 4   | Measurement type                                                         | 4             | 0 or 1            | 0: FFT<br>1: ADC                 |
| 5   | Frequency resolution of a sampling time t depends on measurement type    | 4             | 512               | Hz (1/s) or t (ns)               |
| 6   | Normation                                                                | 4             | 0,1,2             |                                  |
| 7   | Sampling frequency                                                       | 4             | 1953125           | Hz (1/s)                         |
| 8   | ADC Resolution                                                           | 4             | 12                |                                  |
| 9   | Ambient temperature                                                      | 4             | -500...500        | °C                               |
| 10  | Distance between sensor and first object(round-trip-time)                | 4             | 0-2 <sup>32</sup> | t (μs)                           |
| 11  | Time of flight of US in air for and backward--> distance to first object | 4             | 0-2 <sup>32</sup> | t (μs)                           |
| 12  | FFT Window length                                                        | 4             |                   |                                  |
| 13  | FFT Window Offset (index -> freq_min_index)                              | 4             |                   |                                  |
| 14  | Software Version (RP) as a string                                        | 4             | V0.2              |                                  |
| 15  | reserved1                                                                | 4             |                   |                                  |
| 16  | reserved2                                                                | 4             |                   |                                  |
| 17  | Data [...]                                                               | 64            |                   | 2 byte each                      |

The detailed description of the headers is given above in Table 4.1 So, to remove the headers from the raw dataset and process it to make it as an input for the

further processing, an extraction Python code was made to do this task.

- i. Created an extraction program whose main functionality is to remove the first 16 columns from the raw data, which is fed as an input file to this program. The program finally saves the extracted file (after the removal of headers) at the desired directory location.
- ii. For the ease of use, the code creates a GUI, which allows the us to select the raw dataset file in the txt format for processing and an output directory to save the final processed file. Below is the code which does the job as described above. This implementation enables an easy-to-understand interface toward processing raw dataset files, taking into consideration the presence of errors and presenting structured output by means of pandas.

```
import os
import pandas as pd
import tkinter as tk
from tkinter import filedialog, messagebox

def select_input_file():
    input_file = filedialog.askopenfilename(title="Select Input File", filetypes=[("Text files", "*.txt")])
    input_path.set(input_file)

def select_output_directory():
    output_dir = filedialog.askdirectory(title="Select Output Directory")
    output_path.set(output_dir)

def process_file():
    input_file = input_path.get()
    output = output_path.get()
    if not input_file or not output:
        messagebox.showerror("Error", "Please select both an input file and output directory.")
        return
    if not input_file.endswith('.txt'):
        messagebox.showerror("Error", "Please select a valid .txt file.")
        return
    print(f"Processing {input_file}")
    with open(input_file, 'r') as f:
        lines = f.readlines()
    data = []
    for line in lines:
        values = line.split()[16:]
        data.append(values)
    df = pd.DataFrame(data)
    output_file_name = os.path.splitext(os.path.basename(input_file))[0]
    output_file_path = os.path.join(output, f"Extracted_{output_file_name}.txt")
    with open(output_file_path, 'w') as f_out:
        for row in df.iteruples(index=False, name=None):
            f_out.write(' '.join(map(str, row)) + '\n')
    messagebox.showinfo("Done", f"The file {input_file} has been processed!")

root = tk.Tk()
root.title("File Processor")
input_path = tk.StringVar()
output_path = tk.StringVar()
tk.Label(root, text="Input File:").grid(row=0, column=0, padx=10, pady=10)
tk.Entry(root, textvariable=input_path, width=50).grid(row=0, column=1, padx=10, pady=10)
tk.Button(root, text="Browse", command=select_input_file).grid(row=0, column=2, padx=10, pady=10)
tk.Label(root, text="Output Directory:").grid(row=1, column=0, padx=10, pady=10)
tk.Entry(root, textvariable=output_path, width=50).grid(row=1, column=1, padx=10, pady=10)
tk.Button(root, text="Browse", command=select_output_directory).grid(row=1, column=2, padx=10, pady=10)
tk.Button(root, text="Process File", command=process_file).grid(row=2, column=1, padx=10, pady=20)
root.mainloop()
```

Listing 4.1: Code snippet depicting the logic of removing the headers from the raw dataset.

## 4.6. Kalman Filter Model Development

After the data is processed, which basically means the headers are removed from the raw dataset then, the further formatted data will be used in the processing of Kalman Filter. The Kalman filter implementation in this research is tailored for the specific characteristics. Noise filtering of the sensor signals is achieved using a Kalman filter, the sound pressure calculation is performed and the thresholding mechanism including Niblack and Yanowitz-Bruckstein detects the peaks which is further used as input to feature extraction and classification using machine learning algorithms. The workflow of the model is shown in the below Figure 4.8.

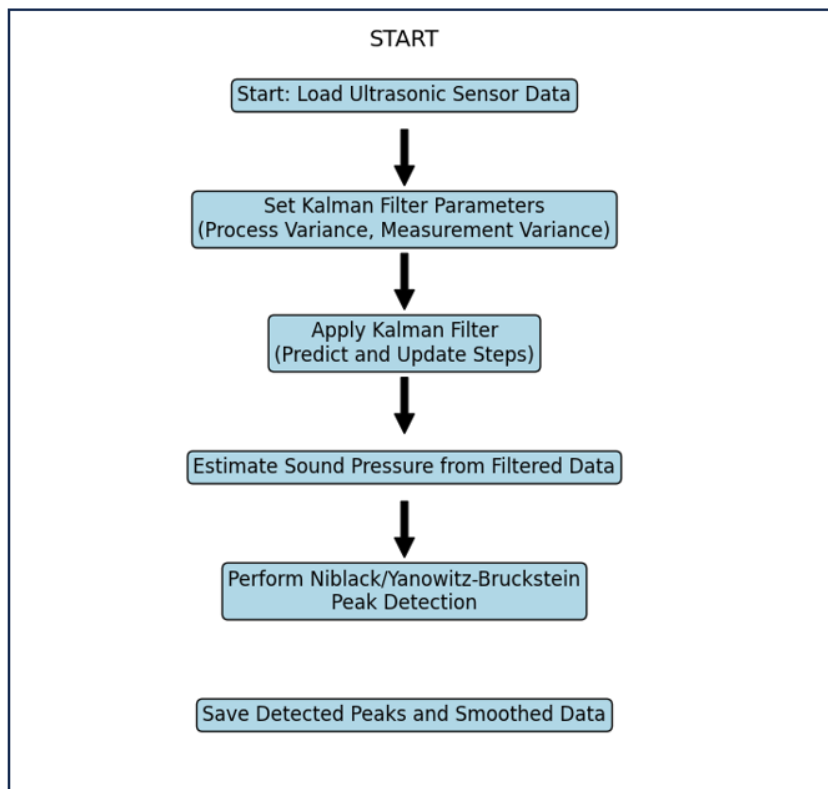


Figure 4.8: Flowchart of Kalman Filter Model

### 4.6.1. Noise Suppression and sound pressure estimation

One of the most important issues when working with data from ultrasonic sensors is handling noise. To address the issue of noise, a Kalman filter is applied to the raw ultrasonic sensor data. A Kalman filter with **state space model** is used to smooth the raw ultrasonic signal.

#### 1. Kalman Filter with State-Space Model

The Kalman filter is based on the state-space model which estimates three main variables: pressure, velocity, and acceleration. It works in two stages: the prediction stage, information about the state transition matrix is included; at every step, the Kalman filter predicts the successive state-namely, pressure, velocity, and acceleration-of the system. In the update stage the filter corrects the prediction using the observation matrix (H) and measurement variance (R) after having the next sensor reading. Some of the key steps of the model includes:

- **State Transition Matrix (A):** Models the progression of the system's state over time, considering relationships between pressure, velocity, and acceleration.
- **Observation Matrix (H):** Maps the state variables (pressure, velocity, and acceleration) to the observed signal.
- **Initial State Estimates (x):** The first sensor reading and initial values for velocity and acceleration are set to zero.
- **Process (Q) and Measurement (R) Covariance Matrices:** These handle uncertainties in the prediction model (process variance) and sensor measurements (measurement variance), respectively.

## 2. Estimated Sound Pressure

Some of the parameters on which sound pressure depends include the following: the distance between the sensor and the reflective surface, the velocity of the sound wave, and the properties of the medium surrounding the system.

Thus, the estimated sound pressure becomes the most key feature that is further used in classification. Occupancy versus non-occupancy status of a chair is determined by the system based on the analysis of changes in sound pressure. Below code-snippet presents Kalman filter with its state-space model:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import tkinter as tk
from tkinter import filedialog, scrolledtext
from scipy.signal import find_peaks
import threading
def kalman_filter_state_model(signal, temperature, sampling_frequency, piston_radius,
    process_variance=1e-4, measurement_variance=1e-1):
    n_iterations = len(signal)
    A = np.array([[1, 1/sampling_frequency, 0.5/(sampling_frequency**2)],
        [0, 1, 1/sampling_frequency],
        [0, 0, 1]])
    H = np.array([[1, 0, 0]])
    x = np.array([signal[0], 0, 0])
    Q = process_variance * np.eye(3)
    R = np.array([[measurement_variance]])
    P = np.eye(3)
    estimated_pressure = np.zeros(n_iterations)
    smoothed_signal = np.zeros(n_iterations)
    speed_of_sound = 331.3 + 0.606 * temperature
    air_density = 1.225
    for t in range(n_iterations):
        x = A @ x
        P = A @ P @ A.T + Q
        z = np.array([signal[t]])
        y = z - H @ x
        S = H @ P @ H.T + R
        K = P @ H.T @ np.linalg.inv(S)
        x = x + K @ y
        P = (np.eye(3) - K @ H) @ P
        distance = calculate_distance(t, temperature, sampling_frequency)
```



```

if distance > 0:
    estimated_pressure[t] = (4 * piston_radius * air_density /
        (distance * speed_of_sound**2)) * x[0]
    smoothed_signal[t] = x[0]
return estimated_pressure, smoothed_signal

```

*Listing 4.2: Code showing the calculation of the estimated sound pressure.*

#### 4.6.2. Niblack and Yanowitz Bruckstein peaks detection

After the noise is suppressed from the raw data and it gets smooth, the next step involved here is to detect the Niblack peak and Yanowitz bruckstein peaks separately. The code here does the job of saving the peaks detected into the separate output files for both the Niblack and Yanowitz bruckstein thresholding methods along with the **500** samples values before as well as after the point of peaks for each sample. The peaks from both thresholding methods are saved in 2 separate files so that it could be used further for the classification using the MLP Classifier.

```

def yanowitz_bruckstein_thresholding(signal, window_size=100,
    threshold_init_factor=0.5, min_peak_prominence=0.1):
    threshold = threshold_init_factor * np.max(signal)
    for i in range(window_size, len(signal) - window_size):
        window = signal[i - window_size:i + window_size]
        local_max = np.max(window)
        if local_max > threshold:
            threshold = local_max - min_peak_prominence
            if signal[i] >= threshold and signal[i] == local_max:
                if all(signal[i] >= signal[j] for j in range(i - window_size, i + window_size) if j != i):
                    return i
    return None
def niblacks_local_thresholding(signal, window_size=2000, k=2.7):
    n = len(signal)
    for i in range(window_size, n - window_size):
        local_segment = signal[i - window_size:i + window_size]
        local_mean = np.mean(local_segment)
        local_std = np.std(local_segment)
        threshold = local_mean + k * local_std
        if signal[i] > threshold:
            if all(signal[i] >= signal[j] for j in range(i - window_size, i + window_size) if j != i):
                return i
    return None

```

*Listing 4.3: Code snippet showcasing the of Niblack and Yanowitz local thresholding*

```

estimated_pressure, smoothed_signal = kalman_filter_state_model(data_row, temperature,
    sampling_frequency, piston_radius)
peak_index_niblack = niblacks_local_thresholding(estimated_pressure)
if peak_index_niblack is not None:
    start_index = max(peak_index_niblack - 500, 0)
    end_index = min(peak_index_niblack + 500, len(data_row))
    self.niblack_data.append(data_row[start_index:end_index])
    self.log_message(f"Row {i}: Niblack peak detected at index: {peak_index_niblack}")
peak_index_yb = yanowitz_bruckstein_thresholding(estimated_pressure)
if peak_index_yb is not None:
    start_index = max(peak_index_yb - 500, 0)
    end_index = min(peak_index_yb + 500, len(data_row))

```

```

self.yanowitz_data.append(data_row[start_index:end_index])
self.log_message(f"Row {i}: Yanowitz-Bruckstein peak detected at index:
{peak_index_yb}")
self.pressure_df.iloc[i] = estimated_pressure
self.update_plot(data_row, smoothed_signal, estimated_pressure, peak_index_yb,
peak_index_niblack)

```

Listing 4.4: Code snippet with logic to detect Niblack and Bruckstein peak and store data

### 4.6.3. Signal Envelope in Kalman Filter Processing

The envelope detection is realized by using the function named **find\_peaks** from the **scipy.signal** library, which looks for the positive peaks in the sound pressure data. Those peaks correspond to the points where the signal has a local maximum. Having detected such peaks, the envelope was created by interpolation between these peaks. Below steps were followed up for the implementation:

1. **Peak Detection:** The following code finds the positions of the positive peaks in the estimated sound pressure signal using a given function called **find\_peaks**.
2. **Interpolation:** Thus, detected peaks have a smooth envelope drawn by interpolating between them.
3. **Envelope Visualization:** After that, an envelope is plotted on top of the sound pressure plot-the envelope underlines visually the borders of the signal and therefore, points out the biggest fluctuation.

```

self.ax2.plot(estimated_pressure, label='Estimated Sound Pressure', color='lightpink')
positive_peaks, _ = find_peaks(estimated_pressure)
if len(positive_peaks) > 0:
    envelope = np.interp(np.arange(len(estimated_pressure)), positive_peaks,
estimated_pressure[positive_peaks])
    self.ax2.plot(envelope, label='Envelope', color='black', linestyle='-')
if peak_index_yb is not None:
    self.ax2.plot(peak_index_yb, estimated_pressure[peak_index_yb], 'rx', markersize=12,
label="Yanowitz-Bruckstein Peak")
if peak_index_niblack is not None:
    self.ax2.plot(peak_index_niblack, estimated_pressure[peak_index_niblack], 'gx',
markersize=12, label="Niblack Peak")

```

Listing 4.5: Code for signal envelope

### 4.6.4. GUI for visualization

A GUI has been made using Python's **Tkinter** library so that the usability and user experience could be enhanced. This will make the work much easier for the user because, through this application, one can interact with the system without necessarily having the user alter anything in the code or running terminal commands. Therefore, such key tasks as loading of data, choosing the parameters, and visual results can be provided in a more intuitive and user-friendly interface. It will help in showing the real time visualization of raw and smooth signal, estimated sound pressure as well as act as a peak detection marker.



Furthermore, after the signal has been processed, the system allows the options for saving the resulting data-particularly the detected peaks and the smoothed signal-for further analysis and for training machine learning models such as the MLP classifier. Outputs of the Niblack and Yanowitz-Bruckstein peak detection methods can be exported into .txt files to keep features extracted for later uses. The peak data is also saved, together with a fully smoothed signal after the Kalman filter is applied. This would allow the user to make some offline analysis of the filtered data, or even use it as an input to other algorithms.

- The below image depicts the built GUI. We will select the extracted file (headers removed) using the browse button.
- The selected file path shows in the bottom space and all the other processing are also displayed there.

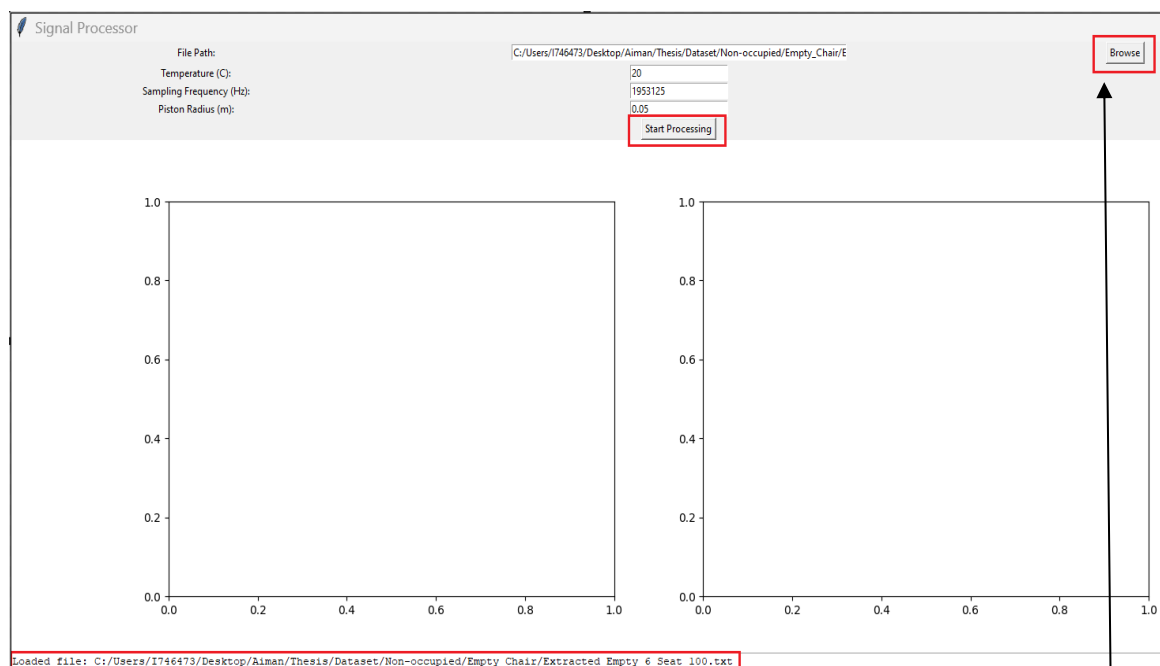


Figure 4.9: GUI for the Kalman Filter Process

File loading as well as processing of each line is displayed here.

'Browse' button to start the file processing.

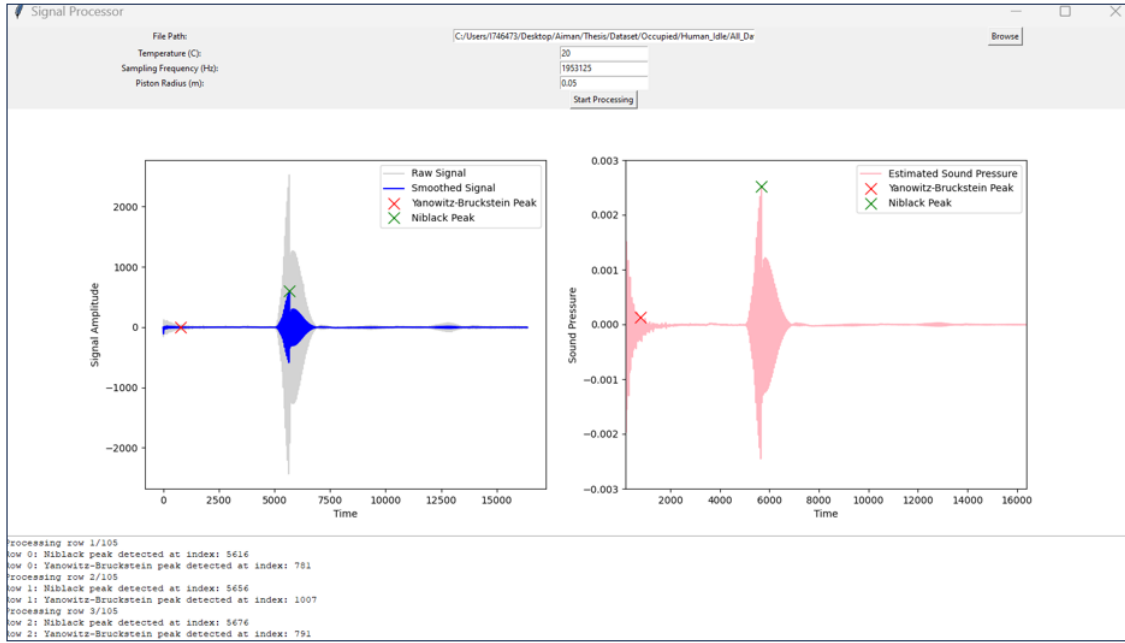


Figure 4.10: GUI depicting Left plot shows the raw signal processing, right plot shows the processed signal after Kalman filtering.

- After processing all the sample values from the input file, finally the GUI asks for saving the data of Niblack Peak as shown in image below. This data consists of total 1000 samples, with a peak value and 500 samples before and after the Niblack peak.
- The same process is adopted for the Yanowitz-Bruckstein thresholding as shown in the below image. This file also consists of 1000 sample data, with the peak value and 500 samples before and after the Yanowitz-Bruckstein peak.

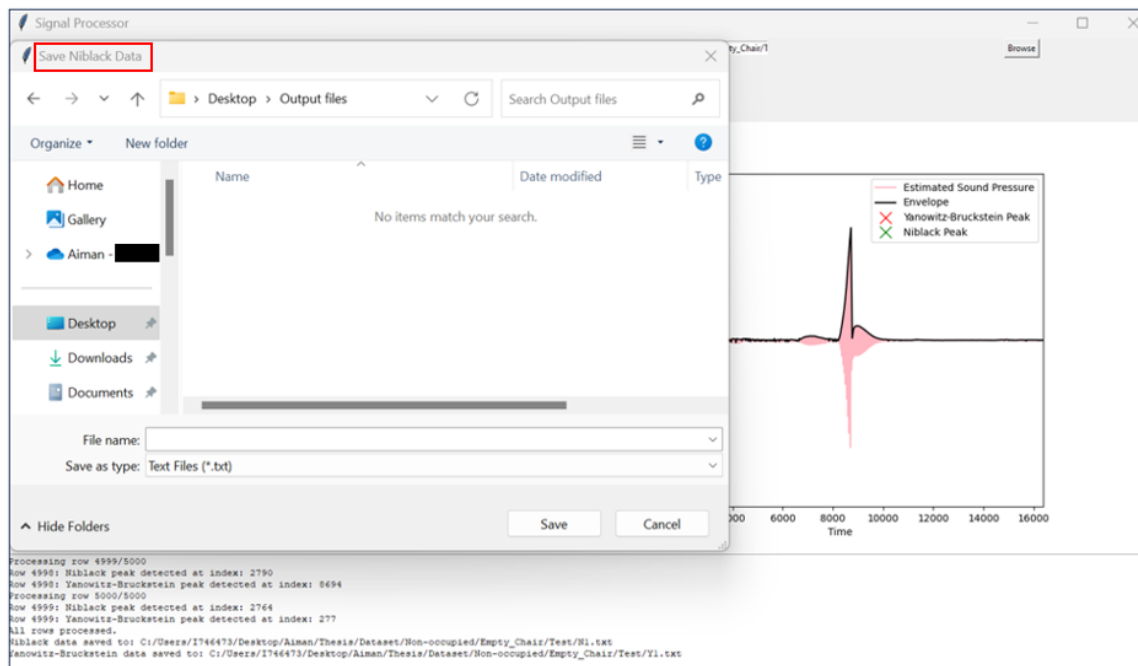


Figure 4.11: GUI showing the option to save the Niblack peak data after the Kalman Filter processing.

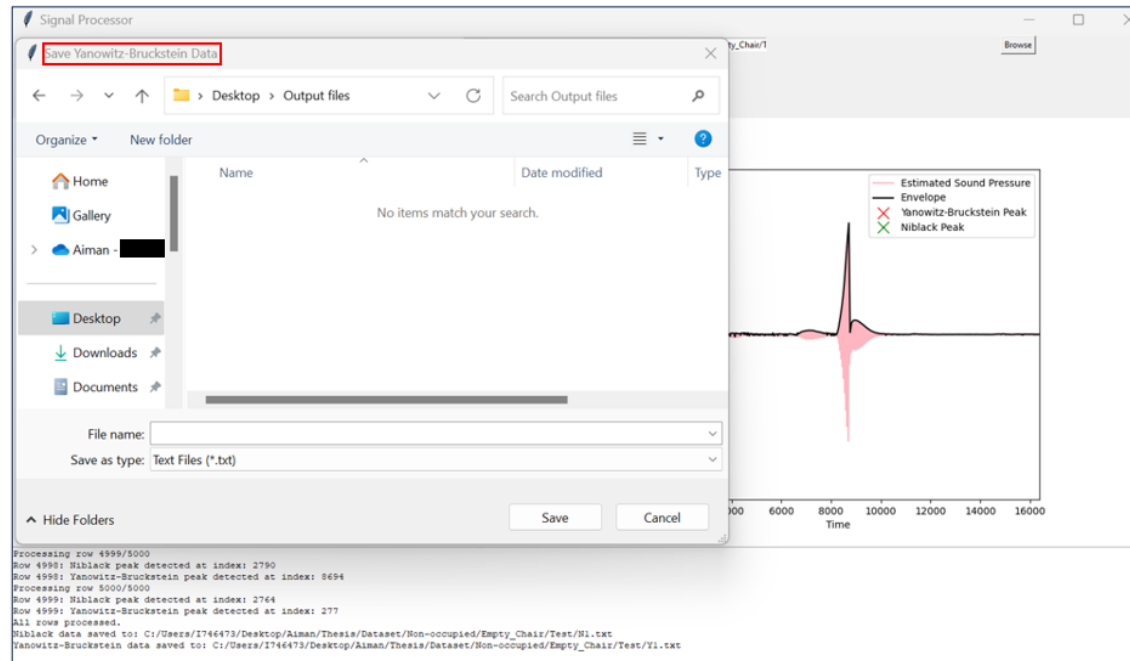


Figure 4.12: GUI showing the option to save the Yanowitz peak data after the Kalman Filter processing.

## 4.7. MLP Classification Model

Now, for classifying the occupancy status of a chair as either 'occupied' or 'non-occupied', the processed ultrasonic data relies on an MLP classifier. The inputs that drive the MLP classifier are from the output files generated by the Kalman filter arising from the above processing. The classifier is trained and tested on data that represents occupied and non-occupied chair states. Below is the flowchart showing how the workflow of an MLP Classifier goes for person detection using ultrasonic data. It describes-from loading and preprocessing, training the model, its evaluation, up to decision making by accuracy thresholds. Below Figure 4.13 shows the detailed representation.

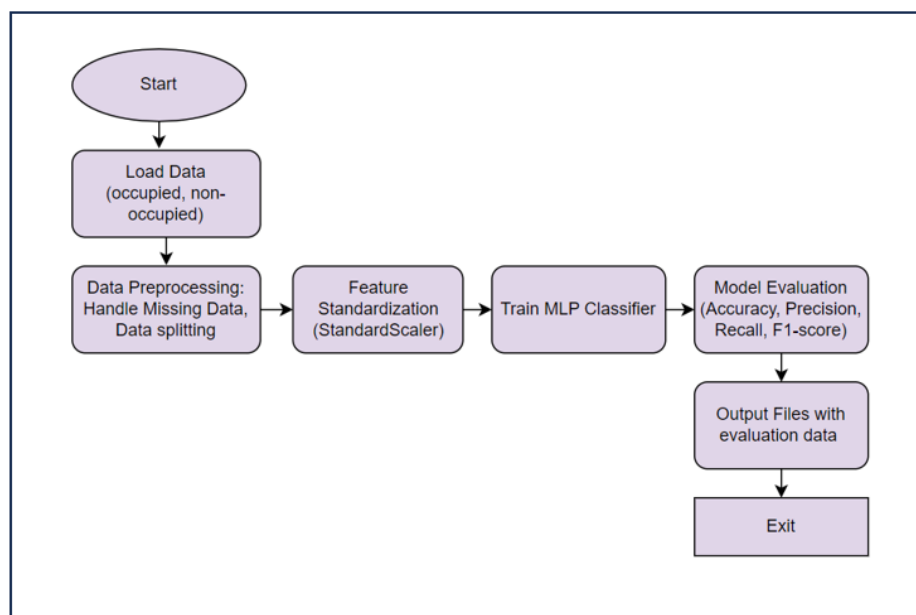


Figure 4.13: Flowchart of the MLP Classification Model

Some of the Major steps includes:

**1. Input Data and Preparation** - After raw ultrasonic data is filtered using the Kalman filter and peak detection, two files are generated namely Niblack\_Peak.txt and Yanowitz\_Peak.txt.

a. **Input file loading:** The file which carries the signal data with the peaks detected by using a Niblack local thresholding method and Yanowitz-Bruckstein peak detection method are used as an input dataset in .txt format. The function shown in the image below reads data from the text file and in case, there are bad lines in the file it skips them and continue reading.

```
def read_data(file_path, delimiter=','):
    print(f'Reading data from: {file_path}')
    try:
        df = pd.read_csv(file_path, delimiter=delimiter, header=None, on_bad_lines='skip',
engine='python')
        print(f'Successfully read file with shape: {df.shape}')
        return df
    except Exception as e:
        print(f'Error reading the file {file_path}: {e}')
        return pd.DataFrame()
```

*Listing 4.6: Code to load the input file*

b. **Save Output files:** The function in the below image saves various results of the MLP classifier such as accuracy, precision, recall, F1-score, specificity etc., to the user selected directory.

```
def save_output_files(directory, y_test, y_pred, cf_matrix, accuracy, class_report, metrics):
    np.savetxt(os.path.join(directory, 'actualValues.txt'), y_test, fmt='%d')
    np.savetxt(os.path.join(directory, 'predictedValues.txt'), y_pred, fmt='%d')
    with open(os.path.join(directory, 'mlp_performance_results.txt'), 'w') as f:
        f.write("Confusion Matrix:\n")
        f.write(np.array2string(cf_matrix) + "\n\n")
        f.write(f"Accuracy: {accuracy:.4f}\n\n")
        f.write("Classification Report:\n")
        f.write(class_report + "\n")
        f.write(f"Precision (PPV): {metrics['precision']:.4f}\n")
        f.write(f"Recall (TPR): {metrics['recall']:.4f}\n")
        f.write(f"F1 Score: {metrics['f1_score']:.4f}\n")
        f.write(f"Specificity (TNR): {metrics['specificity']:.4f}\n")
        f.write(f"True Positive (TP): {metrics['TP']}\n")
        f.write(f"True Negative (TN): {metrics['TN']}\n")
        f.write(f"False Positive (FP): {metrics['FP']}\n")
        f.write(f"False Negative (FN): {metrics['FN']}\n")
        f.write(f"Accuracy: {accuracy:.4f}\n")
    print(f"\nModel performance metrics have been saved successfully to '{directory}'.")
```

*Listing 4.7: Code to save three output files with actual values, predicted values and performance result of matrix.*

- c. **Confusion Matrix Plotting:** This function has been used in the visualization of a confusion matrix with the help of a heat map; this gives better insight into model performance. Here, **cf\_matrix** is used as the input parameter for the confusion matrix.

```
def calculate_metrics(cf_matrix):
    TN = cf_matrix[0][0] # True Negative
    FP = cf_matrix[0][1] # False Positive
    FN = cf_matrix[1][0] # False Negative
    TP = cf_matrix[1][1] # True Positive
    specificity = TN / (TN + FP) if (TN + FP) > 0 else 0
    recall = TP / (TP + FN) if (TP + FN) > 0 else 0 # TPR / Sensitivity
    precision = TP / (TP + FP) if (TP + FP) > 0 else 0 # PPV
    return {
        'specificity': specificity,
        'recall': recall,
        'precision': precision,
        'TN': TN,
        'FP': FP,
        'FN': FN,
        'TP': TP
    }
```

*Listing 4.8: Code logic with the calculation of specificity, recall and precision.*

- d. **Handling Missing Values:** The dataset's missing values will be imputed with the mean in order not to have any problems while training.
- e. **Splitting Data:** The dataset will be divided into two sets-a training set that contains **67%** of the data and a test set containing **33%**-ensuring that the classifier is trained on other unseen examples. The data labelling is done for **1** for non-occupied, **2** for occupied.

```
print("Select Non-occupied (Empty) Data File")
non_occupied_chair_path = askopenfilename(title="Select Non-occupied (Empty) Data File",
filetypes=[("Text Files", "*.txt"), ("All Files", "*.*")])
print(f"Non-occupied file selected: {non_occupied_chair_path}")
print("Select Occupied (Human) Data File")
occupied_chair_path = askopenfilename(title="Select Occupied (Human) Data File",
filetypes=[("Text Files", "*.txt"), ("All Files", "*.*")])
print(f"Occupied file selected: {occupied_chair_path}")
if non_occupied_chair_path and occupied_chair_path:
    non_occupied_df = read_data(non_occupied_chair_path, delimiter=' ')
    occupied_df = read_data(occupied_chair_path, delimiter=' ')
    if non_occupied_df.empty or occupied_df.empty:
        print("Error reading one or both of the data files. Exiting.")
        exit()
    X = pd.concat([non_occupied_df, occupied_df], ignore_index=True)
    y = [1] * len(non_occupied_df) + [2] * len(occupied_df)
    X = X.values
    y = np.array(y)
    imputer = SimpleImputer(strategy='mean')
    X = imputer.fit_transform(X)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

*Listing 4.9: Code with the logic of splitting the data into testing and training sets.*

- 2. Feature Standardization** - The features are to be normalized before the training of the MLP classifier. **StandardScaler** scales features so that their mean would be 0 and standard deviation 1. Hence, the performance of the MLP classifier becomes effective, and learning gets stabilized.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

*Listing 4.10: Code logic for normalization*

- 3. Design and Training** - In this context, for the proposed MLP classifier, one hidden layer containing three neurons and one output layer with class size two-occupied and non-occupied-will be considered. The proposed classifier will be trained using the labeled dataset. Backpropagation and gradient descent shall be used for a maximum of 30,000 iterations with an assurance of sufficient model convergence.

```
clf = MLPClassifier(solver='lbfgs', alpha=1e-3,
                  hidden_layer_sizes=(3,), random_state=42, max_iter=30000)
print("Training the MLP classifier...")
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

*Listing 4.11: Code logic with the maximum iterations*

- 4. Model Testing and Evaluation** - After training the classifier, test sets were used to check the performance of the trained classifier. Evaluation metrics for performance evaluation are as follows:

- 1. Confusion Matrix:** A confusion matrix is drawn indicating correct and incorrect classifications entailing True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).
- 2. Metrics of Accuracy and Classification:** It calculates the overall accuracy of the classifier along with precision, recall, and F1-score, which give insight into the model's ability to correctly identify occupied and non-occupied chairs.
- 3. Visualization:** It then plots a confusion matrix as a heat map to graphically illustrate how many times both correct and incorrect predictions fall under frequency for both classes: occupied and non-occupied.

```
cf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cf_matrix)
plot_confusion_matrix(cf_matrix)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
class_report = classification_report(y_test, y_pred, target_names=['Non-occupied', 'Occupied'])
print("\nClassification Report:")
print(class_report)
metrics = calculate_metrics(cf_matrix)
```

```
metrics['f1_score'] = f1
print(f"\nDetailed Metrics:")
print(f"True Positive (TP): {metrics['TP']}")
print(f"True Negative (TN): {metrics['TN']}")
print(f"False Positive (FP): {metrics['FP']}")
print(f"False Negative (FN): {metrics['FN']}")
print(f"Precision (PPV): {metrics['precision']:.4f}")
print(f"Recall (TPR): {metrics['recall']:.4f}")
print(f"Specificity (TNR): {metrics['specificity']:.4f}")
print(f"F1 Score: {metrics['f1_score']:.4f}")
print(f"Accuracy: {accuracy:.4f}")
```

*Listing 4.12: Code for confusion matrix and other classification parameters*

5. **Saving and Exporting Model Results** - Finally, the results of the classifier are saved for future analysis. The system allows users to:
1. **Save Predictions:** The actual and predicted values are saved in separate files.
  2. **Save Model Performance Metrics:** The confusion matrix, accuracy, precision, recall, F1-score, and other key performance metrics are saved in a text file named as **mlp\_classification\_results** for detailed analysis.

## Chapter 5

# Result and Analysis

In this section, results of the machine learning model developed for the detection of a person in an office environment using ultrasonic sensor data are presented and analyzed. The main aim is to assess how well such a classifier has been configured back on the features selected, and various preprocessing operations performed. Pursuant to this, it is the intention of this research to critically examine a variety of criteria such as accuracy, precision, recall, F1-score, and so forth in order to evaluate how well the model can successfully label the occupied and the situation of non-occupied chair in an office environment.

Apart from the overall evaluation, in this section, a comparison analysis will be drawn for different scenarios such as: The effect of applying a Kalman filter for noise suppression, such as how much this improves the quality of the signal, and the classification accuracy will be analyzed for the raw datasets (without Kalman filter) as well as processed dataset (with Kalman filter).

Moreover, a comparative analysis between two thresholding methods for the occupied chair detection will be done using two different methodologies of feature extraction, namely, Niblack and Yanowitz-Bruckstein, and the performance of the model will be compared to determine which methodology can provide more effective and distinguishable features for occupied chair detection.

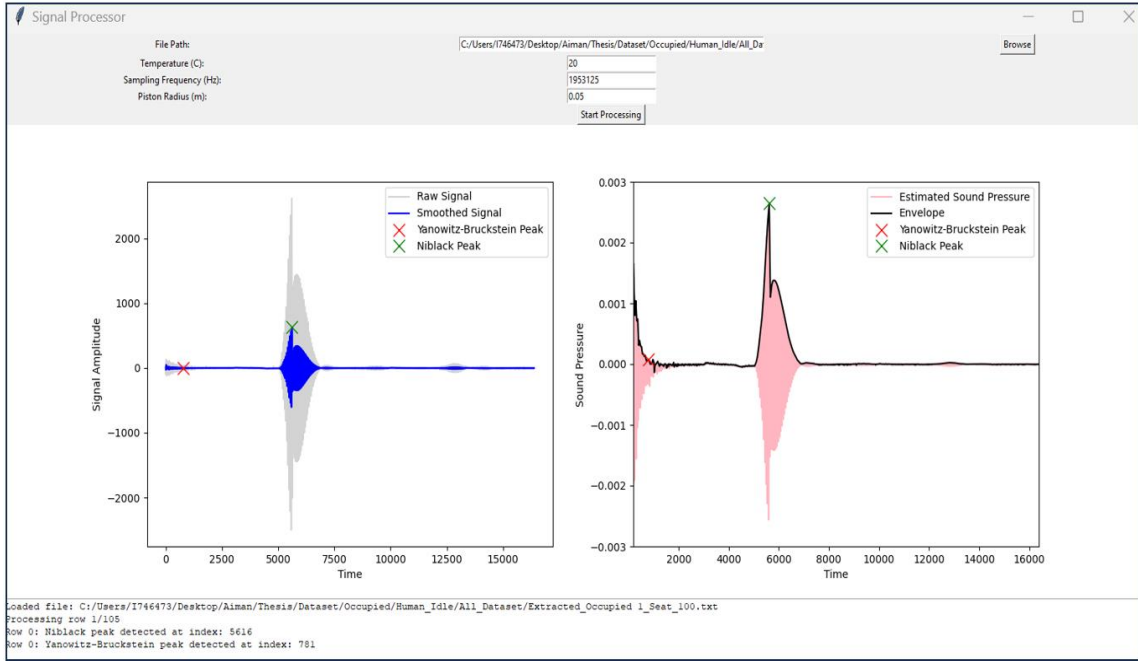
### 5.1. Noise reduction Analysis

The following section gives the comparative analysis of the impact of the Kalman filter on the ultrasonic sensor data, using both Niblack and Yanowitz-Bruckstein peak detection methods for feature extraction:

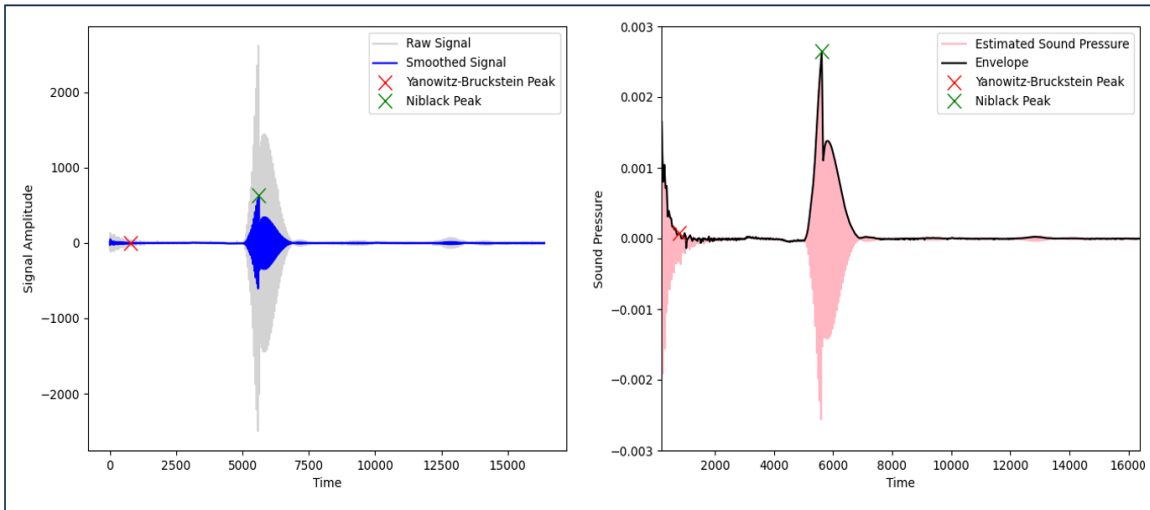
#### 5.1.1. Analysis of occupied dataset

The raw signal in the occupied chair dataset is quite noisy and contains several peaks due to reflections from the human body, as depicted in the attached Figure 5.1. In the occupied chair state, these irregular peaks with increased amplitude in the signal make the detection of meaningful patterns quite difficult. After the Kalman filtering, the signal looks much smoother as there is much less noise, and most of the key reflections are brought out much better. The peaks due to Niblack and Yanowitz-Bruckstein are rather well seen in this case and provide good features for the classification.





a.

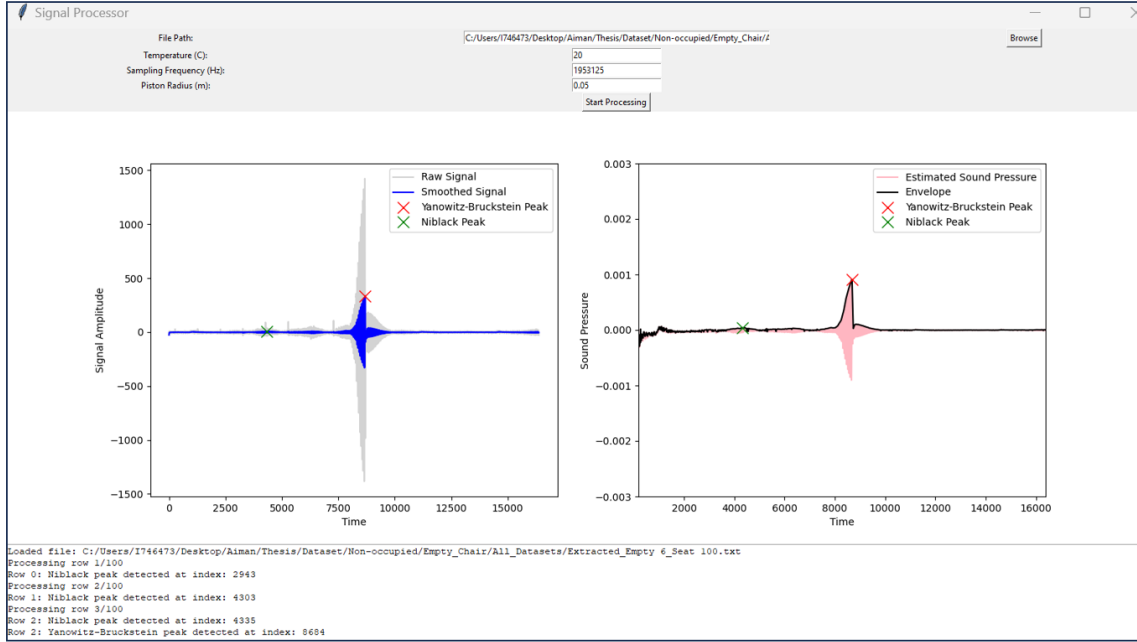


b.

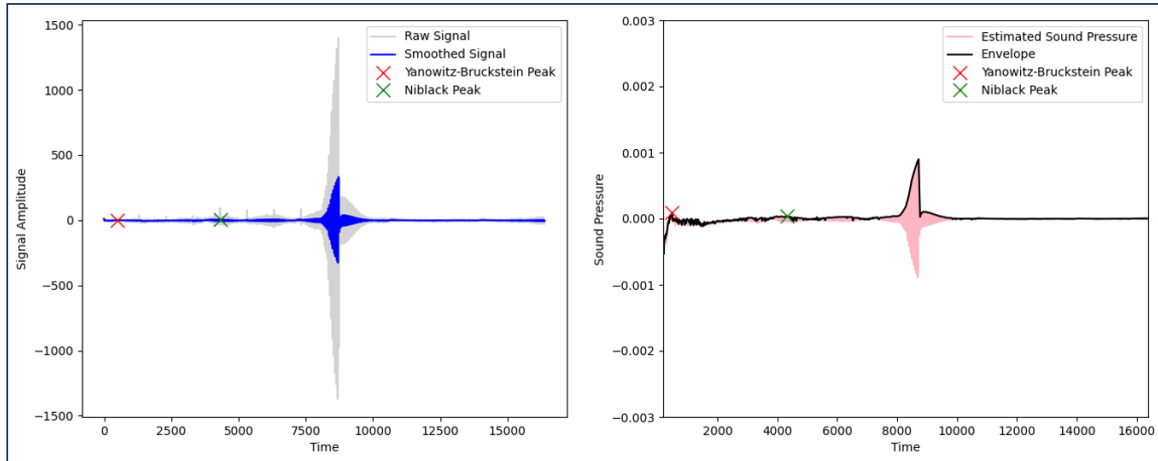
Figure 5.1: Plot (a) and plot (b) showing the signal amplitude along with peaks and envelope for occupied dataset.

### 5.1.2. Analysis of non-occupied dataset

For a non-occupied chair, as can be seen from the Figure 5.2 below, raw signal which is on the left side is much cleaner as there are fewer peaks and fluctuations since there was no human body to give complex reflections. The amplitude of the reflections is much lower, and the structure of the empty chair was much simpler, hence it yielded a cleaner dataset. This signal is further smoothed by the post-processing stage using a Kalman filter, giving a very smooth waveform, as seen in both methods of Niblack and Yanowitz-Bruckstein. After the Kalman filter processing and removal of noise, the graph looks like the one on the right side which is the sound pressure vs time plot. This shows the amplitude after noise reduction with the prominent Niblack and Yanowitz peaks detected at a certain point.



a.



b.

Figure 5.2: Plot (a) and plot (b) showing the signal amplitude along with peaks and envelope for occupied dataset.

### 5.1.3. Visual Comparison after Kalman Filter Application

- While the Kalman filter presents noise reduction in both data sets, the signal is noisier for the occupied chair state; hence, the magnitude of noise reduction is more pronounced in this condition than when the chair is empty.
- In each scenario, the smoothed signal is of higher quality and better suited for the machine learning model to extract significant features such as the reflection peaks.
- Therefore, it is apparent that Niblack and Yanowitz-Bruckstein peaks are the critical indicators of the fact that the Kalman filter succeeded in filtering out the meaningful signal information without dealing with irrelevant noise.

## 5.2. Impact of Kalman Filter on Classification

This section considers the performance analysis of the Kalman filter according to the obtained results from the classifier using both Kalman-filtered data and raw data.

The raw dataset readings (after the header removal) for different configurations of occupied and non-occupied chair were taken. Afterwards, these readings were used as an input to the designed MLP classifier to calculate the accuracy, precision, and other parameters.

On the other hand, the two data files generate after the Kalman filter processing namely Niblack Peak and Yanowitz-Bruckstein peaks were used as an input to the MLP classifier to calculate the accuracy, F1-score etc. as shown in the below tables. The primary aim in the below tables is to show a comparative analysis of the accuracy for all the test cases with and without Kalman filter. This is done to determine the effect of Kalman filter on the accuracy and precision of detection of occupied and non-occupied chairs in an office environment.

This section carries out an analysis of the performance of the classification system for four different cases, considering raw and Kalman-filtered data, using Niblack and Yanowitz-Bruckstein peak detections. To quantify the results presented in this section, accuracy, precision, recall, and F1-score metrics aim to show how well the Kalman filter reduces noise and improves the accuracy of classification. For each test case, below tables will summarize the results under three conditions:

1. Without Kalman filter (raw data).
2. With Kalman filter using **Niblack peak detection**.
3. With Kalman filter using **Yanowitz-Bruckstein peak detection**.

### I. Case 1: Non-Occupied chair (Empty) vs. Occupied chair (Human idle)

This case assesses the classifier performance between an empty chair and a chair occupied by a human who is just sitting idle. In the table below, the Niblack method gave the highest values for each metric-seconded closely by the Yanowitz-Bruckstein method.

*Table 5.1 : Classification parameter table for non-occupied chair (empty) vs. occupied chair (Human Idle)*

| Condition                          | True Positive (TP) | True Negative (TN) | False Positive (FP) | False Negative (FN) | Accuracy | Precision | Recall | F1-score |
|------------------------------------|--------------------|--------------------|---------------------|---------------------|----------|-----------|--------|----------|
| Without Kalman Filter              | 1588               | 1649               | 29                  | 34                  | 0.9809   | 0.9821    | 0.9790 | 0.9809   |
| With Kalman Filter (Niblack Peak)  | 1561               | 1674               | 8                   | 9                   | 0.9948   | 0.9949    | 0.9943 | 0.9948   |
| With Kalman Filter (Yanowitz Peak) | 1377               | 1245               | 87                  | 77                  | 0.9411   | 0.9406    | 0.9470 | 0.9411   |

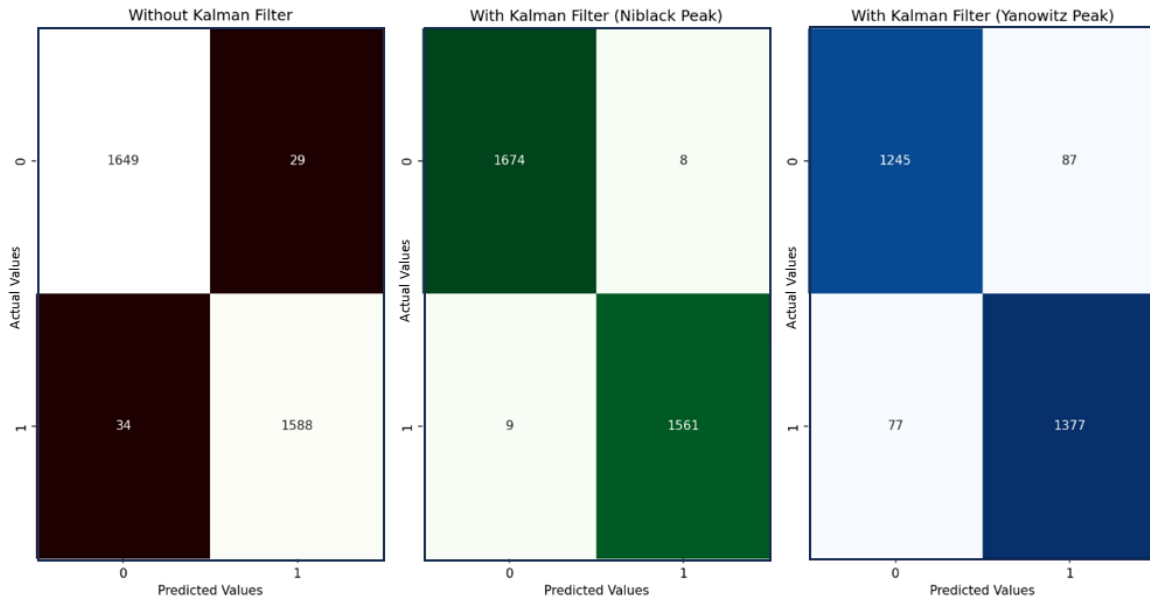


Figure 5.3: Confusion matrix for case I

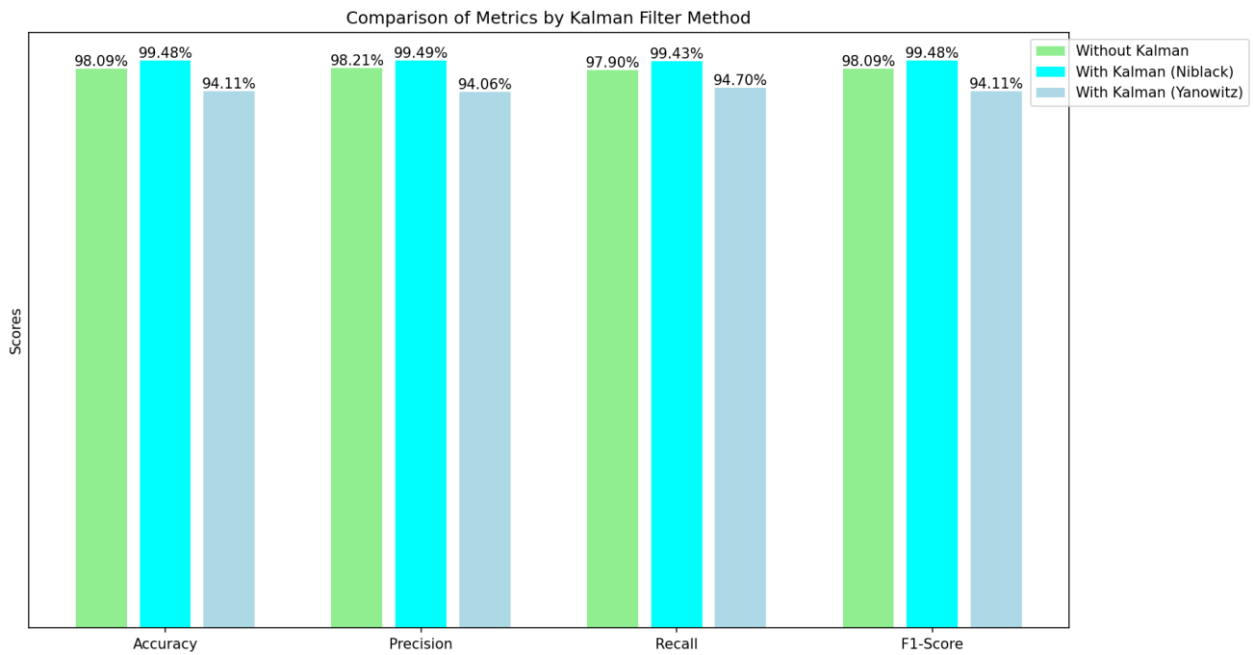


Figure 5.4: Bar chart for the comparison of matrices for case I.

The Kalman filter application, especially with peaks from Niblack, greatly improved the performance of classification compared to raw data. Without a filter, the classifier typically shows higher misclassification rates, while after the addition of the Kalman filter with peak detection according to Niblack, the number of misclassifications noticeably declined and true positives hugely raised along with overall accuracy. The Yanowitz peak method yields improvement, although it is slightly less effective than Niblack in the reduction of errors generated by noise. Accuracy, precision, recall and F1-score of the Kalman filter with Niblack peak is comparatively higher than that of the Yanowitz peak.

## II. Case 2: Non-Occupied chair (Boxes) vs. Occupied chair (Human idle)

In this scenario, we will look at the classifier's ability to differentiate between a chair that has cardboard boxes placed on it and one that is occupied by a human sitting idle on the chair. In this case, the outcome without Kalman filter approach gave the best results in terms of precision and recall, but both the Kalman-filtered approaches presented accuracy relatively close to each other. This setup, with the boxes on an empty chair, can result in reflections not that different from an occupied chair. The Kalman filter is very effective in noise reduction and could easily suppress the relevant signal variations associated with an occupied state. That shows that the Kalman filtering may be less ideal for all types of settings, especially for environments having complex, non-human reflectors, such as boxes.

Table 5.2: Classification parameter table for non-occupied chair (Boxes) vs. occupied chair (Human Idle)

| Condition                          | True Positive (TP) | True Negative (TN) | False Positive (FP) | False Negative (FN) | Accuracy | Precision | Recall | F1-score |
|------------------------------------|--------------------|--------------------|---------------------|---------------------|----------|-----------|--------|----------|
| Without Kalman Filter              | 1565               | 1668               | 15                  | 52                  | 0.9797   | 0.9905    | 0.9678 | 0.9797   |
| With Kalman Filter (Niblack Peak)  | 1455               | 1678               | 4                   | 115                 | 0.9634   | 0.9973    | 0.9268 | 0.9634   |
| With Kalman Filter (Yanowitz Peak) | 1392               | 1449               | 56                  | 74                  | 0.9562   | 0.9613    | 0.9495 | 0.9562   |

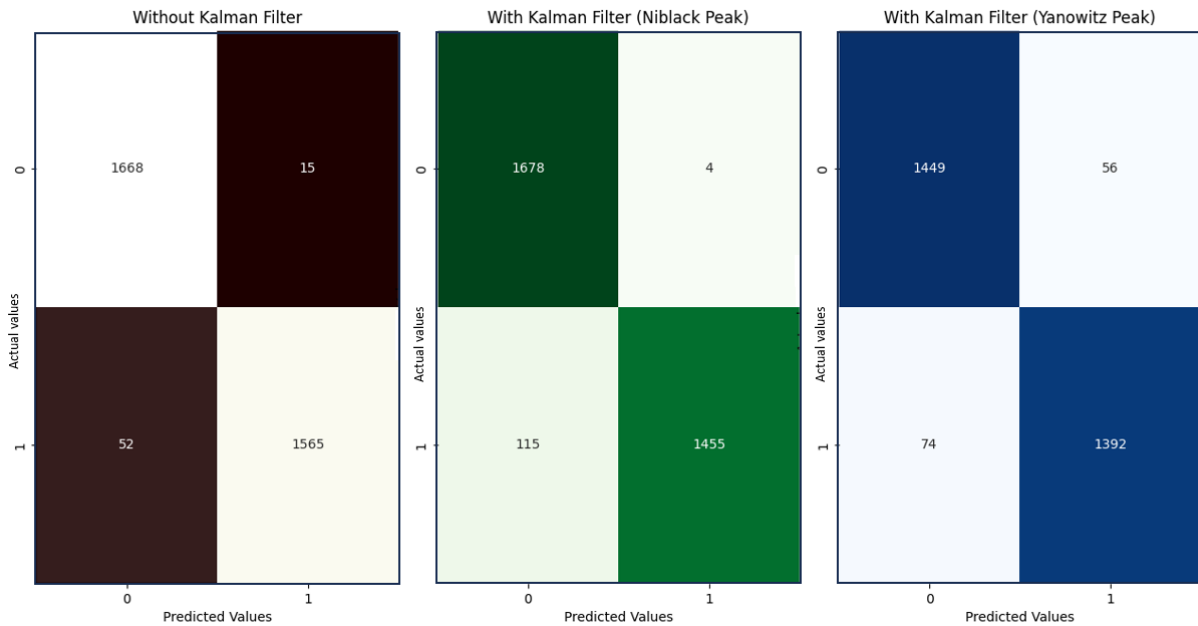


Figure 5.5: Confusion matrix for case II

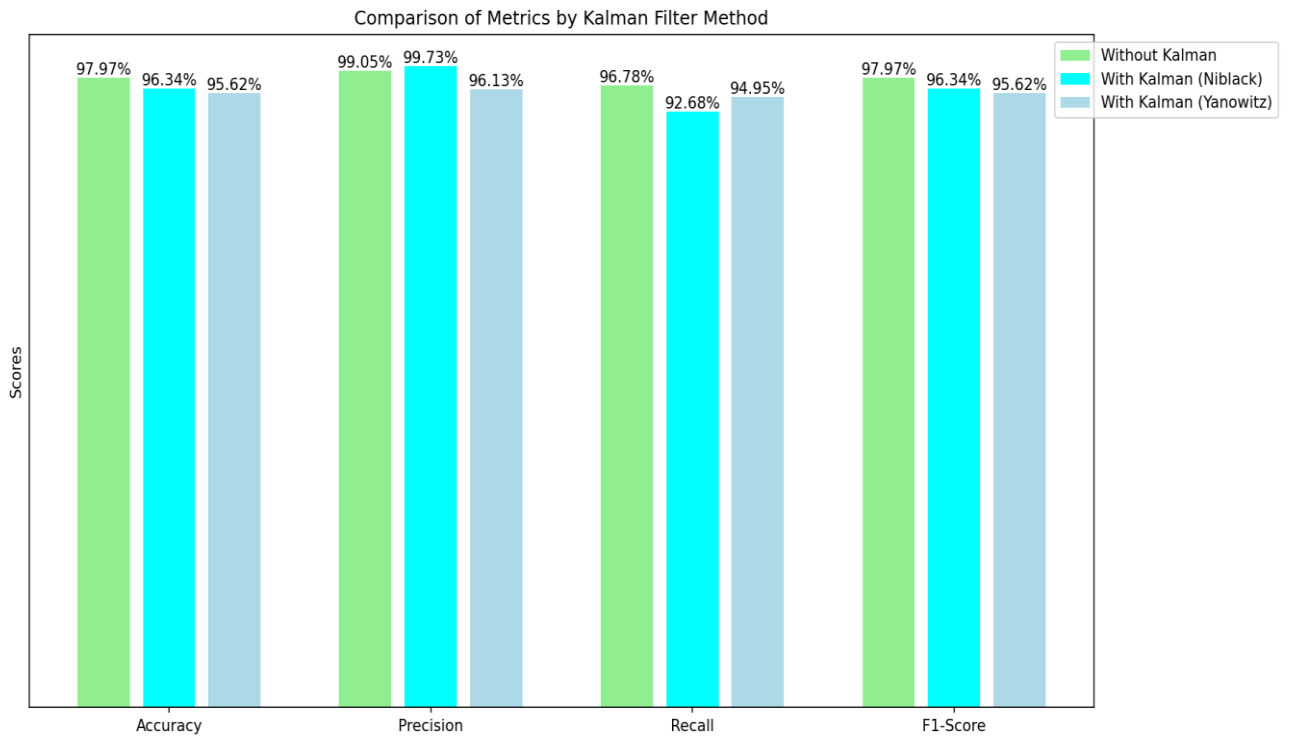


Figure 5.6: Bar chart for the comparison of matrices for case II.

### III. Case 3: Non-Occupied chair (Empty) vs Occupied chair (Human moving)

This case represents the performance of the classifier in differentiating between an empty chair and a chair occupied by a human who is moving around—for example, moving his head, or gesturing with his hands. As can be guessed, the motion adds more complexity to the classification process. The raw data had a high accuracy of 98.24%, but this case showed greater differences between raw and Kalman filtered results.

Moreover, While the Niblack peak detection gives the highest percentage accuracy of 99.57% and precision of 99.94%, it consolidates that the Kalman filter has been efficient in handling the movement noise. Yanowitz-Bruckstein peak detection also showed better performance but was not as effective compared to the Niblack method, with relatively lower precision and recall.

Table 5.3: Classification parameter table for non-occupied chair (Empty) vs. occupied chair (Human moving)

| Condition                          | True Positive (TP) | True Negative (TN) | False Positive (FP) | False Negative (FN) | Accuracy | Precision | Recall | F1-score |
|------------------------------------|--------------------|--------------------|---------------------|---------------------|----------|-----------|--------|----------|
| Without Kalman Filter              | 1590               | 1654               | 28                  | 30                  | 0.9824   | 0.9827    | 0.9815 | 0.9824   |
| With Kalman Filter (Niblack Peak)  | 1564               | 1678               | 1                   | 13                  | 0.9957   | 0.9994    | 0.9918 | 0.9957   |
| With Kalman Filter (Yanowitz Peak) | 1367               | 1212               | 99                  | 99                  | 0.9287   | 0.9325    | 0.9325 | 0.9287   |

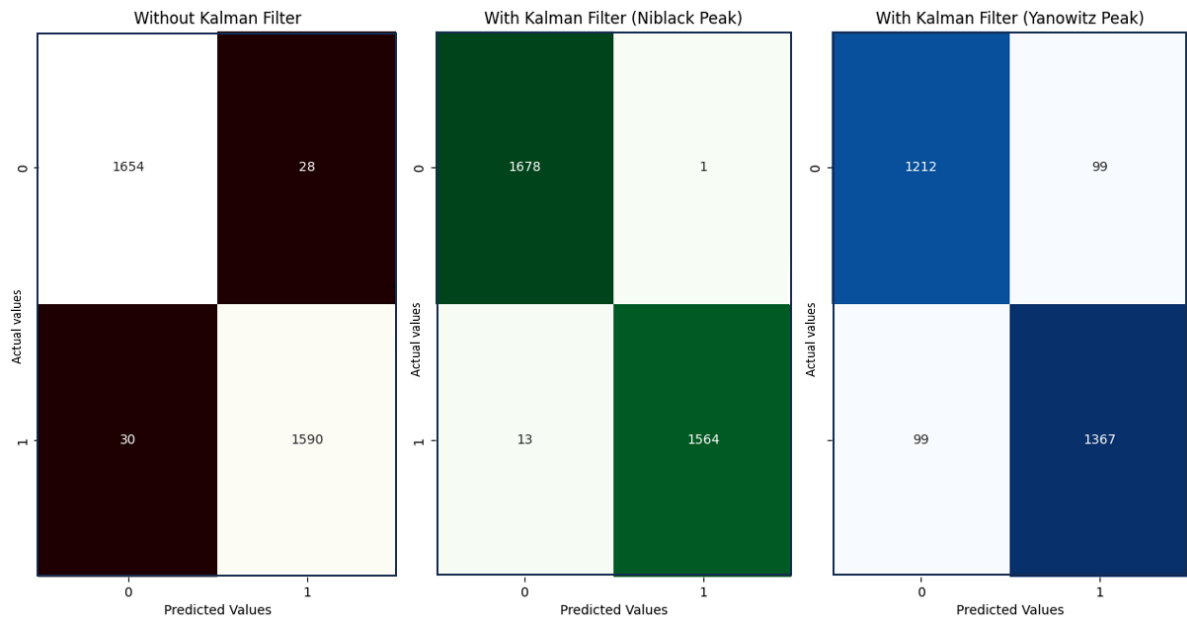


Figure 5.7: Confusion matrix for case III

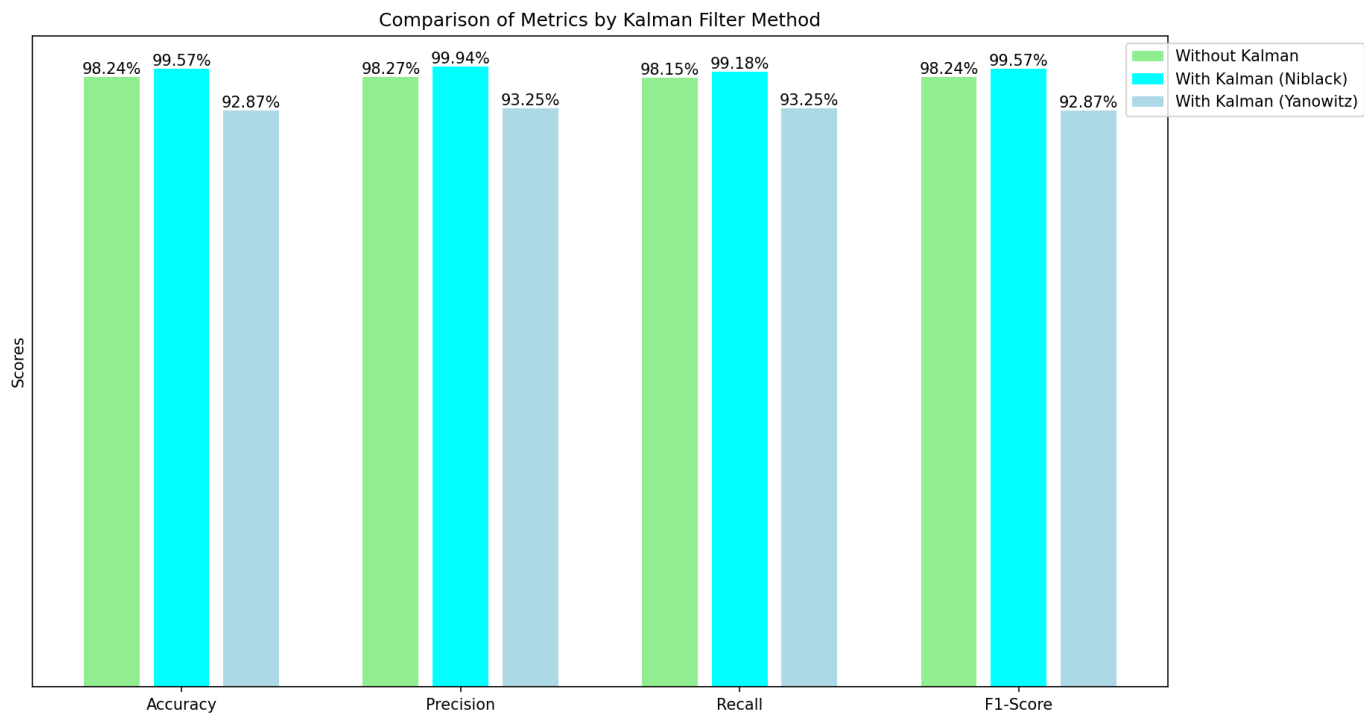


Figure 5.8: Bar chart for the comparison of matrices for case III

#### 4. Case 4: Non-Occupied chair (Boxes) vs. Occupied chair (Human moving)

In this fourth case, the classifier is tested to distinguish between a chair occupied by a human in motion from that of a chair with cardboard boxes placed on it. Movement and other physical objects add greater complexity to the system. The raw data condition saw generally decent performance by the classifier, coming in with 98.88% accuracy. The Niblack peak detection is consistent, but its accuracy is slightly lower compared to the raw data, which is 98.16%, and lower in recall.

Table 5.4: Classification parameter table for non-occupied chair (Boxes) vs. occupied chair (Human moving)

| Condition                          | True Positive (TP) | True Negative (TN) | False Positive (FP) | False Negative (FN) | Accuracy | Precision | Recall | F1-score |
|------------------------------------|--------------------|--------------------|---------------------|---------------------|----------|-----------|--------|----------|
| Without Kalman Filter              | 1593               | 1671               | 18                  | 19                  | 0.9888   | 0.9888    | 0.9882 | 0.9888   |
| With Kalman Filter (Niblack Peak)  | 1546               | 1650               | 29                  | 31                  | 0.9816   | 0.9816    | 0.9803 | 0.9816   |
| With Kalman Filter (Yanowitz Peak) | 1392               | 1478               | 30                  | 62                  | 0.9689   | 0.9789    | 0.9574 | 0.9689   |

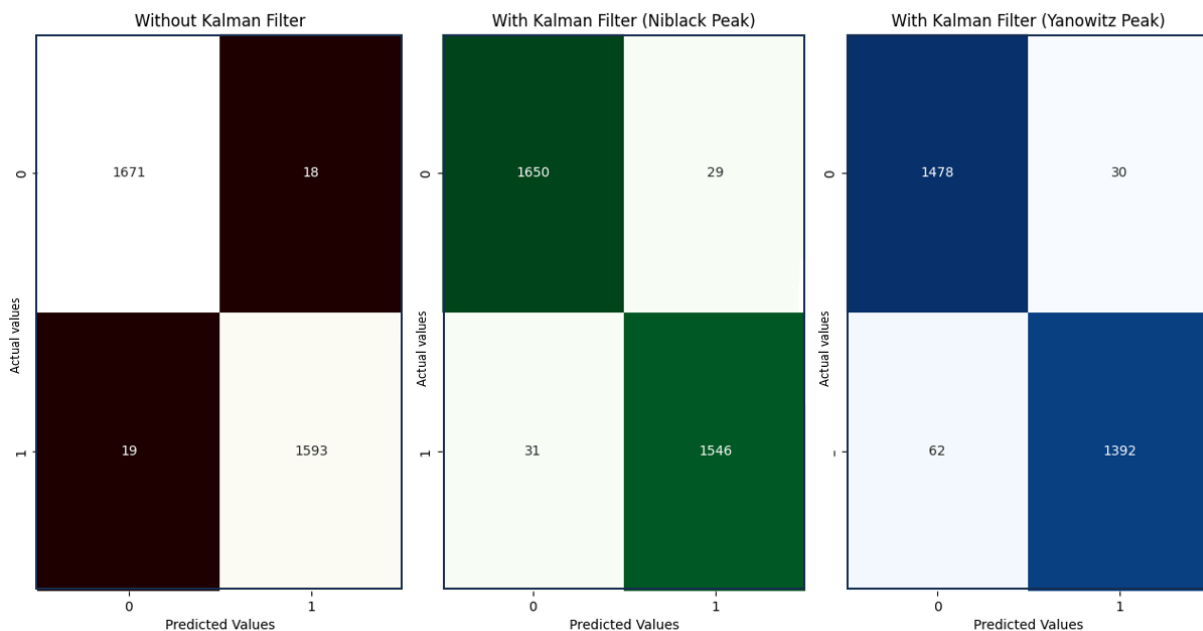


Figure 5.9: Confusion matrix for case IV



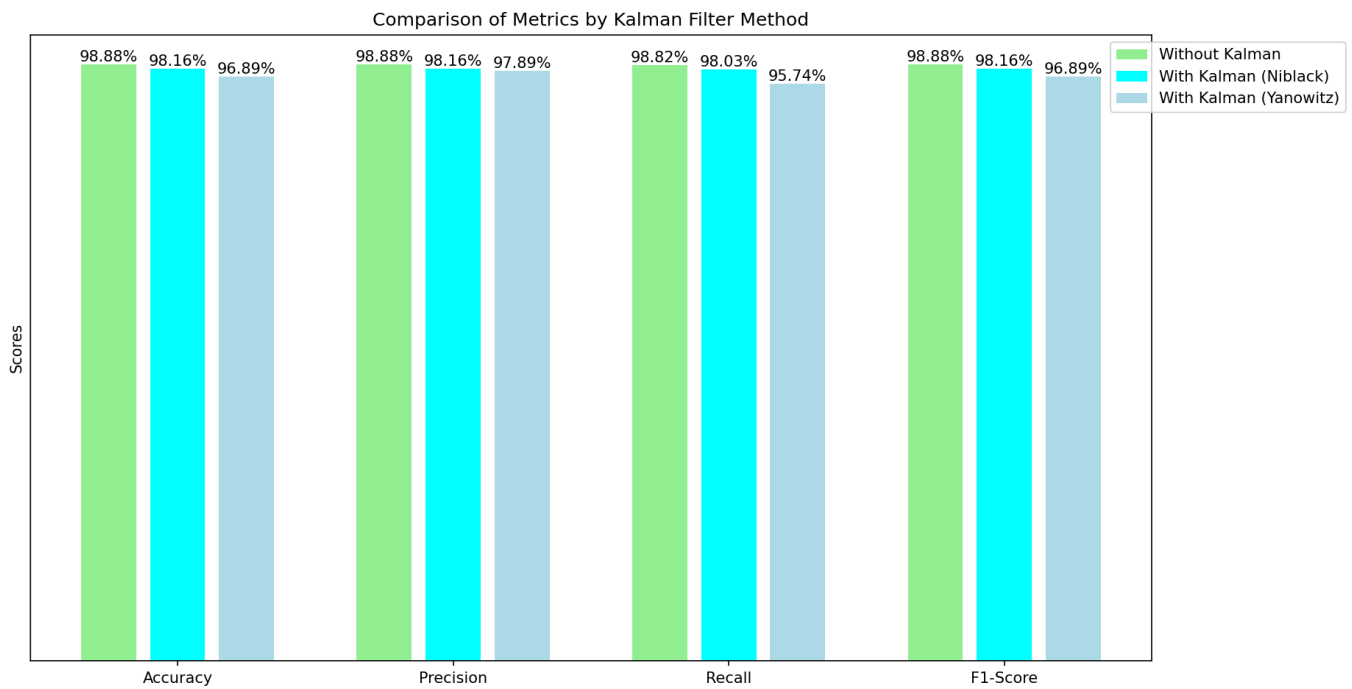


Figure 5.10: Bar chart for the comparison of matrices for case IV

### 5.3. Comparative Performance Analysis

The performance of Niblack Peak and Yanowitz Peak data, as well as the effectiveness of Kalman filtering is compared in this section.

#### 5.3.1. Effectiveness of Kalman Filtering

Indeed, as can be seen by generally higher precision scores in all cases, Kalman filter application-both with Niblack and Yanowitz peaks-improved results by reducing noise. This was in support of the hypothesis that the Kalman filter increases the accuracy of classification because the smoothing of the signal cleans the noise.

#### 5.3.2. Comparative Analysis of Niblack vs. Yanowitz Peak Detection

Overall, the peaks of Niblack tended to report higher accuracy, precision, and recall in case an idle human being exists. That would infer that the nature in which Niblack's method carries out peak detection is more sensitive to changes of subtle signals of a surrounding environment in stationarity, thus reducing false negatives.

Moreover, Yanowitz peaks were undetermined, as both the accuracy and recall became lower in more dynamic cases such as Case 3 and Case 4. Evidently, the Yanowitz approach was less suited to cases that included human movement or challenging, multi-object scenarios; this is because the method fundamentally relies on detecting prominent peaks, which may not capture such nuanced movements.

### 5.3.3. Performance Metrics Evaluation

The performance parameters across all the three different cases of without Kalman filter, with Kalman filter (Niblack peak), with Kalman filter (Yanowitz peak).

- **Accuracy Analysis:** The Accuracy for the classifier showed a significant improvement with Kalman filter, especially in the cases of Niblack peak detection. Without the filter, accuracy was quite stable under simple static conditions, but showed not so good results under the dynamic scenarios that include movement or obstacles. The Niblack peak method had the highest accuracy among all cases, showing its robust noise suppression and adaptability against various occupancy scenarios. Yanowitz peak detection with Kalman filtering also resulted in a higher increase in accuracy than that of raw data, which tended to generally underperform as compared to the Niblack method in dynamic environments.
- **Precision Analysis:** Precision results also showed that using a Kalman filter proved to be useful, particularly when applied with the Niblack peak detection method. In case of without Kalman filter condition, precision was high in scenarios with static conditions, that does not have much of environmental disturbances. While on the other hand precision was less for the cases with increased noise as raw data often leads to false positives.
- **F1-score:** The F1-score is a balance between precision and recall, and its maximum value was reached using the Kalman filter together with Niblack peak detection. Although the raw data had quite reasonable F1-scores in static conditions, it was harder to reach such a balance without noise suppression in more challenging scenarios. Moreover, the Niblack peak detection maintains high F1-scores across all cases by maintaining false positives and missed detections. While the Yanowitz peak detection also improved F1-scores over raw data, its performance was less consistent under some circumstances.

## 5.4. Summary and Perspective

The focus of this study was to optimize the classifier for the detection of a person in an office environment. Red Pitaya based ultrasonic sensor was used to collect the data for the research which used python-based GUI for data processing. This work utilized the Kalman filter and adaptive, as well as local, thresholding methods, like the Yanowitz-Bruckstein method and the method by Niblack, to filter out noise. The calculation of sound pressure level was also performed, and peak data were stored for further analysis using a MLP based machine learning classifier.

This study showcased the Performance metrics results which reveals that Kalman filter combined with Niblack peak gives a balanced approach in the minimization of false positives and missed detections across various cases. On the other hand, the Yanowitz peak method improved classification accuracy, as compared to raw data, but its results varied for dynamic environments, hence requiring adaptable peak detection strategies when ultrasonic sensors are used for real-time applications such as detection of person in an office environment.

### **Future Scope of Improvement**

- The parameters based on the Kalman Filter could be more finely tuned according to the environmental conditions to improve the noise handling and more precise peaks detection.
- Other advanced classifiers such as Convolutional Neural Networks (CNNs), Support Vector Machines (SVM's) or Recurrent Neural Networks (RNNs), could provide more accuracy performance metrics such as accuracy and precision especially in the time sequence data.
- More precise testing could be done under the real-world circumstances such as taking the readings under the real office environment condition with multiple chairs, monitors, and multiple people sitting inside a cabin. The environmental condition could be more optimized according to the different test case scenarios.

## References

1. (Online). <https://robot-electronics.co.uk/htm/srf02tech.htm>
2. (Online). <https://redpitaya.com/>
3. NB Share. (n.d.). *Learn And Code Confusion Matrix With Python*. Retrieved from <https://www.nbshare.io/notebook/626706996/Learn-And-Code-Confusion-Matrix-With-Python>
4. Aggarwal, S. (2023). *Machine Learning algorithms, perspectives, and real-world application: Empirical evidence from United States trade data*. MPRA Paper. Retrieved from <https://mpra.ub.uni-muenchen.de/116579/>
5. Aldahiri, A., Alrashed, B., & Hussain, W. (2021, 03). Trends in Using IoT with Machine Learning in Health Prediction System. 3, pp. 181-207. doi:10.3390/forecast3010012
6. Babu, S., Kumar, C., & Kumar, R. (2006). Sensor Networks for Tracking a Moving Object using Kalman Filtering. In *2006 IEEE International Conference on Industrial Technology* (pp. 1077-1082). doi:10.1109/ICIT.2006.372308
7. Dagang, W. (n.d.). Essential Math for Machine Learning: Confusion Matrix, Accuracy, Precision, Recall, F1-Score. Retrieved from <https://medium.com/@weidagang/demystifying-precision-and-recall-in-machine-learning-6f756a4c54ac>
8. Dehghan-Niri, Ehsan, F., Alireza, S., & Salvatore. (2013, 08). Nonlinear Kalman Filtering for acoustic emission source localization in anisotropic panels. *Ultrasonics*, 54. doi:10.1016/j.ultras.2013.07.016
9. Farhat, R., Mourali, Y., Jemni, M., & Ezzedine, H. (2020). An overview of Machine Learning Technologies and their use in E-learning. In *2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA)* (pp. 1-4). doi:10.1109/OCTA49274.2020.9151758
10. Franklin, W. (2020). The Kalman Filter. Retrieved from <https://thekalmanfilter.com/kalman-filter-explained-simply/>
11. Hauptmann, P., Hoppe, N., & Puettmer, A. (2001). Ultrasonic sensors for process industry. In *2001 IEEE Ultrasonics Symposium. Proceedings. An International Symposium (Cat. No.01CH37263)* (pp. 369-378 vol.1). doi:10.1109/ULTSYM.2001.991644
12. Hubbard, A. (n.d.). *Kalman Filter for State Space Models*. Retrieved from [https://cran.r-project.org/web/packages/kalmanfilter/vignettes/kalmanfilter\\_vignette.html](https://cran.r-project.org/web/packages/kalmanfilter/vignettes/kalmanfilter_vignette.html)
13. Ilya, B., Alfred, B., & Ron, K. (2006). Efficient computation of adaptive threshold surfaces for image binarization. 39(0031-3203), pp. 89-101. doi:<https://doi.org/10.1016/j.patcog.2005.08.011>
14. J., A., H., L., R., E., M., M., & A., J. (2021). Occupancy detection in non-residential buildings

- A survey and novel privacy preserved occupancy monitoring solution. *Emerald Insight*, 17(2634-1964). Retrieved from <https://www.emerald.com/insight/content/doi/10.1016/j.aci.2018.12.001/full/html#abstract>
- 15. Jaisawal, S. (n.d.). Retrieved from <https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning>
- 16. Jost, D. (n.d.). Retrieved from <https://www.fierceelectronics.com/sensors/lidwave-lands-funding-fuel-4d-lidar-development>
- 17. Junjing Yang, M., & Santamouris, S. (2016). Review of occupancy sensing systems and occupancy modeling methodologies for the application in institutional buildings. *Energy and Buildings*, 121(0378-7788), 344-349. doi:<https://doi.org/10.1016/j.enbuild.2015.12.019>
- 18. Kailai Sun, Q., & Zhao, J. (2020). A review of building occupancy measurement systems. *Energy and Buildings*, 216(0378-7788), 109965. doi:<https://doi.org/10.1016/j.enbuild.2020.109965>
- 19. Kamthe, A., Jiang, L., Dudys, M., & Cerpa, A. (2009, 02). SCOPES: Smart Cameras Object Position Estimation System. (978-3-642-00223-6), pp. 279-295. doi:10.1007/978-3-642-00224-3\_18
- 20. Khurshid, K., Siddiqi, I., Faure, C., & Vincent, N. (2009, 01). Comparison of Niblack inspired Binarization Methods for Ancient Documents. pp. 1-10. doi:10.1117/12.805827
- 21. Kurniawati, A., Suprpto, Y., & Yuniarno, E. (2020). Multilayer Perceptron for Symbolic Indonesian Music Generation. In *2020 International Seminar on Intelligent Technology and Its Applications (ISITIA)* (pp. 228-233). doi:10.1109/ISITIA49792.2020.9163723
- 22. Michael, G., & Myeongsu, K. (2018). *Prognostics and Health Management of Electronics: Fundamentals, Machine Learning, and the Internet of Things*. Wiley-IEEE Press.
- 23. Mittal, R., Mongia, S., & Chugh, U. (2022). Machine Learning Techniques Towards Efficient Occupancy Detection for Industrial Urban Spaces. In *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)* (Vol. 1, pp. 322-326). Faridabad, India. doi:10.1109/COM-IT-CON54601.2022.9850523
- 24. Mittal, S., Prasad, T., Saurabh, S., Xue Fan, & Hyunchul Shin. (2012). Pedestrian detection and tracking using deformable part models and Kalman filtering. In *2012 International SoC Design Conference (ISOCC)* (pp. 324-327). doi:10.1109/ISOCC.2012.6407106
- 25. Mörzinger, R., Thaler, M., Stalder, S., Grabner, H., & Van Gool, L. (2011). Improved person detection in industrial environments using multiple self-calibrated cameras. In *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (pp. 486-491). doi:10.1109/AVSS.2011.6027381
- 26. Mustafa, W., Khairunizam, W., Ibrahim, Z., A. B., S., & Razlan, Z. (2018). A Review of Different Segmentation Approach on Non Uniform Images. In *2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA)* (pp. 1-6). doi:10.1109/ICASSDA.2018.8477611

27. N., T., & Gupta, R. (2020). A Survey on Machine Learning Approaches and Its Techniques. In *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)* (pp. 1-6). doi:10.1109/SCEECS48394.2020.190}
28. Nabaee, M., Pooyafard, A., & Olfat, A. (2008). Enhanced object tracking with received signal strength using Kalman filter in sensor networks. In *2008 International Symposium on Telecommunications* (pp. 318-323). doi:10.1109/ISTEL.2008.4651321
29. Niblack, W. (1986). *An introduction to digital image processing*. Prentice-Hall. doi:<https://worldcat.org/title/710027746>
30. Nimmada, G., & Basha, S. (2024, 04). A Novel Method for Predicting the Flight Price using Multilayer Perceptron Algorithm Compared with Support Vector Machine Algorithm. 1-5. doi:10.1109/ICONSTEM60960.2024.10568803
31. Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 62-66. doi:10.1109/TSMC.1979.4310076
32. Patkar, A., & Tasgaonkar, P. (2016). Object recognition using horizontal array of ultrasonic sensors. In *2016 International Conference on Communication and Signal Processing (ICCSP)* (pp. 0983-0986). doi:10.1109/ICCSP.2016.7754294
33. Puthanpurayil,, T. (2023). *Automatic Labelling System using ML*. doi:<https://github.com/TiniyaVinod/Automatic-Labelling-System-using-ML>
34. rolling robot. (2021). Retrieved from [rollingrobotonline.com: http://www.rollingrobotsonline.com/arduino-ultrasonic-sensor](http://www.rollingrobotsonline.com/arduino-ultrasonic-sensor)
35. S.D., Y., & A.M., B. (1989). A new method for image segmentation. *Computer Vision, Graphics, and Image Processing*, 46(0734-189X), 82-95. doi:[https://doi.org/10.1016/S0734-189X\(89\)80017-9](https://doi.org/10.1016/S0734-189X(89)80017-9)
36. Tao, S., Kudo, M., Nonaka, H., & Toyama, J. (2012, 01). Person Authentication and Activities Analysis in an Office Environment Using a Sensor Network. *Communications in Computer and Information Science*, 277. doi:10.1007/978-3-642-31479-7\_19
37. Thomas Rincy , N., & Gupta, R. (2020). A Survey on Machine Learning Approaches and Its Techniques. In *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)* (pp. 1-6). doi:10.1109/SCEECS48394.2020.190
38. Turing, J. (n.d.). *Advanced Time Series Analysis: State Space Models and Kalman Filtering*. Retrieved from <https://janelleturing.medium.com/advanced-time-series-analysis-state-space-models-and-kalman-filtering-3b7eb7157bf2>
39. Vafeiadis, T., Z., S., S., G., I., D., K., S., T., & D., M. (2017). Machine Learning Based Occupancy Detection via the Use of Smart Meters. In *2017 International Symposium on Computer Science and Intelligent Controls (ISCSIC)* (pp. 6-12). doi:10.1109/ISCSIC.2017.15
40. Vaidehi, V., Vasuhi, S., Ganesh, K., Theanammai, C., Babu, N., Uthiravel, N., . . . Chandra, G. (2010). Person tracking using Kalman Filter in Wireless Sensor Network. In *ICoAC 2010* (pp. 60-65). doi:10.1109/ICOAC.2010.5725362

41. Vasavi, S., Nikhita , S., & Sai Krishna, P. (2024). GUI-Enabled Boundary Regularization System for Urban Buildings Using the Tkinter. In *2024 2nd International Conference on Device Intelligence, Computing and Communication Technologies (DICCT)* (pp. 424-429). doi:10.1109/DICCT61038.2024.10532910
42. Wang, B., & Zhang, W. (2023, 07). Research on Edge Network Topology Optimization Based on Machine Learning. pp. 41-46. doi:10.1109/ICAML60083.2023.00018
43. Xiong, Y. (2012). Building text hierarchical structure by using confusion matrix. In *2012 5th International Conference on BioMedical Engineering and Informatics* (pp. 1250-1254). doi:10.1109/BMEI.2012.6513202
44. Yang, Y. (2017). A Signal Theoretic Approach for Envelope Analysis of Real-Valued Signals. *IEEE Access*, 5, 5623-5630. doi:10.1109/ACCESS.2017.2688467