Energy Saving Intelligent Smart Lights/Switches.

Course Information Technology

Individual Project

By Dr Andreas Pech

Jenny Nadar
Mat No: 1427226
jenny.nadar@stud.fra-uas.de

***Abstract*—This study introduces an innovative approach to develop an algorithm that will act as an energy saving intelligent smart lighting system utilizing ADC data from ultrasonic sensor (FIUS). Unlike conventional motion-sensor based lighting, which often fails to accommodate scenarios where individuals remain stationary for extended periods, our system offers a nuanced solution that enhances user convenience while minimizing energy consumption. This system uses ultrasonic sensing systems through the integration of machine learning algorithms. This report delves into an experimental study that leverages the capabilities of ultrasonic sensors, specifically the Red Pitaya STEM Lab board and the Ultrasonic Sensor SRF02, to detect human presence and motion dynamics, ensuring that lights remain active even when slightest human motion is detected. The research employs a multifaceted signal analysis methodology, integrating Ultrasonic Sensors (FIUS) for distance calculation and surface characterization, alongside advanced signal processing techniques such as FFT and the Hilbert Transform. The core of the analysis involves using ML algorithms for distance measurement using Multilayer Perceptron (MLP) and classification of motion and steady environment using Rain Forest algorithm. These models were trained on ultrasonic signal data to classify object types and measure distances accurately. Additionally, enhancements to the ultrasonic measurement system are outlined, focusing on decision speed, accuracy, and usability improvements through optimized algorithms and advanced signal processing.**

Keywords—Red Pitaya, Ultrasonic Sensor SRF02, Fast Fourier Transform, Machine Learning, Supervised Learning, Convolutional Neural Networks, Random Forest, XGBoost.

I. INTRODUCTION

In today's evolving world of technology and energy-conscious world, there is a demand for intelligent solutions that reduce wasting of energy while maintaining user convenience has grown substantially. One of the key areas where this demand is evident is in the lighting systems. Traditional motion sensor-based lights, designed to automatically turn on or off based on movement, help conserve energy but come with significant limitations. Specifically, these systems fail to accommodate scenarios where a person remains stationary, leading to unnecessary deactivation of the lights. This can cause user inconvenience in environments such as offices, homes or public spaces where people may remain still for extended period.

This report introduces an advanced energy-saving intelligent smart lighting system that addresses these limitations by leveraging ultrasonic sensors, specifically FIUS sensor in conjunction with the SRF02 ultrasonic sensor. This system detects human presence more accurately by using ultrasonic wave reflections to measure distance and monitor occupancy, ensuring that lights remain on as long as human presence is detected, regardless of motion. The use of machine learning algorithms further enhances the system's capability to distinguish between occupied and unoccupied environments, optimizing energy use without compromising user comfort.

The core functionality of this system involves processing the raw sensor data to accurately detect human presence and estimate distances. Signal analysis plays a pivotal role in this process. To extract meaningful insights from the ultrasonic sensor data, we apply the Fast Fourier Transform (FFT) to convert the time-domain signals into the frequency domain. This transformation enables the identification of crucial frequency-based features such as mean, root mean square (RMS), maximum value, variance, energy, first peak frequency, and magnitude. These frequency-domain features provide a detailed representation of the ultrasonic signals, capturing both large-scale and subtle variations that correspond to human presence or absence.

In addition to frequency-domain analysis, the time-domain features, such as the change in distance and time of flight, are also crucial in detecting dynamic movements and positions. These combined features create a robust dataset that improves the accuracy of classification and distance estimation.

To ensure the reliability and accuracy of the sensor data, preprocessing techniques are applied to clean and refine the signals. The Hilbert Transform is used for signal envelope detection, which helps eliminate noise and improve the quality of the data. This preprocessing step is critical in ensuring that the features extracted from the sensor data are accurate and representative of the underlying physical events.

Once the signals are pre-processed and analysed, machine learning algorithms are employed to extract features and classify the sensor data. The Multi-Layer Perceptron (MLP) algorithm is used to calculate distances based on the ultrasonic sensor data. Additionally, classification of human presence is performed using the Random Forest algorithm, which excels in distinguishing between various occupancy states based on both time-domain and frequency-domain features. The combination of signal processing techniques and machine learning algorithms ensures high accuracy in determining when lights should remain on or turn off, thus optimizing energy efficiency without sacrificing functionality.

This report will provide a comprehensive overview of the system's design, including the data acquisition process, signal processing techniques such as FFT and Hilbert Transform, and machine learning methodologies for feature extraction, distance calculation, and classification. Experimental results demonstrate the system's superior performance in maintaining lighting only when necessary, making it a compelling solution for energy-efficient smart lighting.

II. METHODOLOGY

The theoretical background of the experiment is mentioned in the above section which consists of the description of the Ultrasonic Red Pitaya sensor, FFT data analysis, Machine Learning algorithm background.

A. *Ultrasonic sensor and Red Pitaya Measurement Board*

A test and measurement board called Red Pitaya STEM Lab [1] is based on a system-on-a-chip (SoC) [2] from the former company Xilinx. It may be configured to function as an oscilloscope, spectrum analyser, LCR meter, or network analyser and can be remotely controlled. The Ultrasonic Sensor SRF02 [3], a single transducer ultrasonic rangefinder in a tiny footprint PCB, was utilized in this configuration. The minimum range of the SRF02 is greater than that of other dual transducer rangers since it only employs one transducer for transmission and receiving. The smallest measuring range is approximately 15 cm (6 inches). With a 5V grounded power supply, it can operate. The Red Pitaya device makes it possible to wirelessly transfer data to a laptop for additional processing.

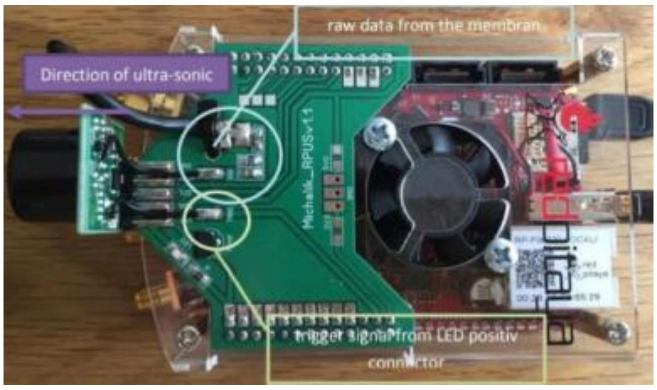

raw data from the membran
Direction of ultra-sonic
trigger signal from LED positiv connector

Fig.1. Ultrasonic sensor and red pitaya device

The sensor is operated under the GNU/Linux operating system. On a computer or mobile device, they can be used to manage and record measurements. In addition to 16 standard input and output ports, the main Red Pitaya unit incorporates two analog RF inputs and outputs. A micro-SD card slot, an RJ45 socket for Ethernet, a USB port, and a micro-USB connector for the console are also included on the board. Radio frequency transmissions can both be received and transmitted by the Red Pitaya which operates in the frequency range of 50MHz.

B. *Theory of Ultrasonic Object Differentiation*

Object differentiation with ultrasound involves using ultrasonic waves to distinguish between different objects or materials based on their physical properties, such as density, composition, and structure. This technique is commonly used in various fields, including medical imaging, non-destructive testing, and industrial inspection.

Ultrasound works on the principle of sending high-frequency sound waves into a material or object and analysing the echoes that bounce back. When an ultrasonic wave encounters a boundary between two different materials (e.g., air and tissue, metal, and plastic), part of the wave is reflected back while the rest continues to penetrate deeper into the material. By analysing the time, it takes for the echoes to return and their amplitude, it's possible to determine the properties of the materials and differentiate between them.

Physical Phenomena and Their Impacts:

1. Attenuation: Attenuation refers to the gradual loss of intensity of the ultrasound signal as it travels through a medium. This phenomenon occurs due to absorption, scattering, and reflection of the sound waves by the material. Attenuation can vary depending on factors such as the frequency of the ultrasound waves, the properties of the material, and the distance travelled. High levels of attenuation can reduce the reliability and precision of ultrasonic object differentiation, as it may result in weaker echoes and decreased signal-to-noise ratio [4].

2. Reflection and Refraction: When an ultrasound wave encounters a boundary between two materials with different acoustic impedances (the product of density and sound velocity), part of the wave is reflected back, and part is transmitted into the second material. The angle of reflection and refraction depends on the acoustic properties of the materials involved. Reflection and refraction can affect the accuracy of object differentiation, especially at interfaces between materials with significant differences in acoustic impedance [5].

3. Scattering: Scattering occurs when ultrasound waves interact with small irregularities or inhomogeneities within a material, causing the wavefront to deviate from its original path. Scattering can result in diffuse reflections and a loss of coherence in the received signal. In materials with high levels of scattering, such as biological tissues, object differentiation may be more challenging due to the complexity of the echo patterns.

4. Multipath Propagation: Multipath propagation refers to the phenomenon where ultrasound waves travel along multiple paths between the transmitter and receiver, leading to the reception of multiple echoes from the same object. This can complicate the interpretation of the received signal and make it more challenging to differentiate between objects, especially in environments with complex geometries or structures [6].

5. Noise: Noise sources, such as electronic interference, environmental factors, and system artifacts, can introduce additional signals into the received ultrasound data, leading to false detections or reduced signal quality. Minimizing noise and optimizing signal processing algorithms are essential for improving the reliability and precision of object differentiation with ultrasound.

In summary, while ultrasound offers valuable capabilities for object differentiation, various physical phenomena such as attenuation, reflection, scattering, multipath propagation, and noise can impact the reliability and precision of the technique. Understanding these phenomena and their effects is crucial for developing effective ultrasonic sensing systems and interpreting ultrasound data accurately in practical applications.

C. Theoretical Framework and Signal Analysis Methodology for Ultrasonic Sensor Systems

This research explores the effectiveness of Frequency-Modulated Integrated Ultrasonic Sensors (FIUS) in distinguishing between inanimate objects and humans by examining the ultrasonic backscatter signals. It's predicated on the notion that the unique surface features of different entities alter the reflected ultrasonic signals in distinct ways, enabling their identification based on the characteristics of these signals. The FIUS sensor works by emitting ultrasonic waves, typically around 40 kHz, and analysing the frequency spectrum of the signals that bounce back from the target. The variation in the reflected signals, influenced by the material and texture of the target's surface, results in discernible changes in the spectrum of the received signals. When these ultrasonic waves are emitted, their interaction with the target's surface generates a backscattered signal that encodes the surface attributes [7].

Surface Characterization Through Signal Analysis:

- *Signal Modulation by Surface Texture:* The study theorizes that smooth surfaces, such as car bumpers, reflect ultrasonic waves more uniformly, resulting in a backscattered signal with a smoother frequency spectrum. In contrast, more complex surfaces, like clothing on pedestrians, cause diffraction and scattering of the ultrasonic waves, leading to a frequency spectrum with multiple peaks and a less uniform distribution.

- *Frequency Shift Analysis:* The analysis focuses on detecting shifts in the peak frequencies of the backscattered signal. A shift from the mean frequency of the emitted signal indicates interaction with non-uniform surfaces, providing a basis for distinguishing between different types of objects.

- *Spectral Shape Examination:* Beyond frequency shifts, the overall shape of the signal's frequency spectrum provides additional insights into the surface's texture. The study posits that non-Gaussian spectral shapes are indicative of complex surfaces, such as those encountered in pedestrian clothing.

The research methodology involves a comparative analysis of the frequency spectra from the backscattered signals off known static and dynamically moving targets. This comparison is intended to uncover specific spectral patterns and markers linked to various surface types. Statistical methods are employed to measure these differences and validate the spectral features' effectiveness as indicators of surface texture [7].

The anticipated results of this theoretical investigation are set to enhance the comprehension of how ultrasonic signals interact with different surfaces, aiding the development of more refined and precise pedestrian detection technologies. By exploiting the subtle variations in ultrasonic backscatter, this approach seeks to improve object recognition capabilities, potentially benefiting a wide range of applications [7].

D. Short Literature overview on ultrasonic distance measurement

Ultrasonic sensors find extensive applications in distance measurement owing to their dependability, precision, and adaptability. These sensors employ ultrasonic waves, characterized by frequencies beyond the upper threshold of human hearing, usually exceeding 20 kHz, to ascertain the distance between the sensor and an object. An overview of distance measurement methods using ultrasonic sensors:

1) Time-of-Flight (ToF) Principle:

Basic Concept: Ultrasonic sensors emit a burst of ultrasonic waves, and the time it takes for the waves to travel to the target object and back is measured. The Time-of-Flight (ToF) principle is a method commonly used in various applications, particularly in the field of distance measurement and imaging. This principle relies on measuring the time it takes for a signal or wave to travel from a source to a target and back again.

In the context of distance measurement, such as with ToF sensors or LiDAR (Light Detection and Ranging) systems, the ToF principle involves sending a signal, often a light pulse or an ultrasonic wave, toward a target. The sensor then measures the time it takes for the signal to travel to the target and be reflected to the sensor. [8]

Distance Calculation: The distance is calculated using the formula:

$$distance = \frac{time\ taken\ x\ speed\ of\ sound}{2}$$

Accuracy: ToF-based ultrasonic sensors can provide high accuracy in distance measurements.

2) Pulse-Echo Method:

Working Principle: Ultrasonic sensors generate short ultrasonic pulses and measure the time it takes for the pulse to travel to the object and return as an echo.

In ultrasonic testing, such as in medical imaging or non-destructive testing of materials, the Pulse-Echo method involves the following steps:

- *Pulse Transmission:* A short burst of ultrasonic waves is generated and directed toward the object or material being examined.
- *Reflection:* When these waves encounter a boundary or interface within the material (due to a change in acoustic impedance), a portion of the waves reflects towards the source.
- *Echo Reception:* A sensor or receiver detects the reflected waves, known as echoes, and measures the time it takes for them to return.
- *Distance Calculation:* The distance to the reflecting surface can be determined by multiplying the time of flight by the speed of sound in the material and dividing by two, as the signal travels to the object and back. [9]

Transducer: The sensor typically consists of a transducer that functions as both a transmitter and a receiver.

Application: Commonly used in industrial automation, robotics, and obstacle detection systems.

3) Phase Shift Measurement:

Basic Concept: The distance is measured by the phase shift between the transmitted and received ultrasonic waves. [10]

Advantages: This method can provide high resolution and is less affected by environmental conditions.

4) Multi-Echo Detection:

Working Principle: Utilizes multiple echoes from different surfaces to improve accuracy and reliability. [10]

The process of multi-echo detection typically includes the following steps:

- Pulse Transmission: A burst of ultrasonic waves is transmitted towards the target or object whose distance is to be measured.
- Echo Reception: The ultrasonic waves encounter surfaces or boundaries within the environment, leading to multiple echoes being reflected toward the sensor.
- Time-of-Flight Measurement: The time it takes for each echo to return to the sensor is measured. Using the time-of-flight principle, distances to the various reflecting surfaces can be calculated.
- Echo Analysis: Multiple echoes may be received due to reflections from different surfaces or objects. The sensor distinguishes between these echoes based on their respective time delays or time-of-flight measurements.
- Distance Calculation: The distances to each reflecting surface are calculated individually using the time-of-flight measurements. This information provides a comprehensive understanding of the spatial configuration of the environment.

Applications: Particularly useful in environments with multiple reflective surfaces or in applications where accurate measurements are critical.

5) Dual-Transducer Systems:

Setup: Involves separate transducers for transmitting and receiving ultrasonic waves. [9]

Benefits: Reduces the impact of signal crossover and enhances performance in challenging conditions.

6) Temperature Compensation:

Challenge: Ultrasonic wave speed is affected by temperature variations. [8]

Solution: Sensors may incorporate temperature sensors to compensate for temperature-induced changes in the speed of sound.

E. Hilbert Transform:

This method analyses the analytic signal obtained from the Hilbert transform of the data to detect peaks. It is effective for detecting peaks with varying widths and shapes. Hilbert Transform is a mathematical operation which provides a way to analyse the phase and amplitude of a complex signal. It is named after the German mathematician David Hilbert and used widely all over the world in signal domain now. The Hilbert transform can be a filter which simply shifts phases of all frequency components of its input by -π/2 radians [11]. The Hilbert transform of a function f(x) is defined by:

$$F(t) = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{f(x)}{t-x}dx$$

Computationally it can be one can write the Hilbert transform as the convolution:

$$F(t) = \frac{1}{\pi t}\cdot f(t)$$

Which, by the convolution theorem of Fourier transforms, may be evaluated as the product of the transform of f(x) with -i*sgn(x), where:

$$\operatorname{sgn}(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

The scientific computing software MATLAB has a Hilbert () function that “computes the so-called discrete-time analytic signal X = Xr + i*Xi such that Xi is the Hilbert transform of Xr” [11]. In MATLAB’s implementation of the Hilbert () function takes advantage of the fast Fourier transform (FFT).

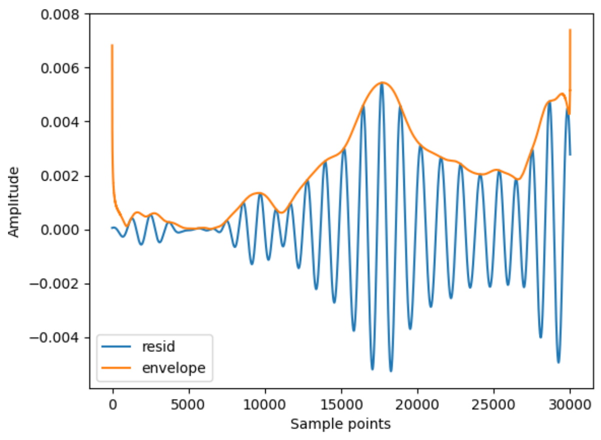


Fig.2. Upper enveloping by Hilbert Transform

The calculation is carried out in three phases by the Hilbert () function - Start by doing the FFT of Xr. Then, make all of the FFT elements that correspond to frequency $-\pi < \omega < 0$ to zero. The last step would be performing the Inverse FFT [12].

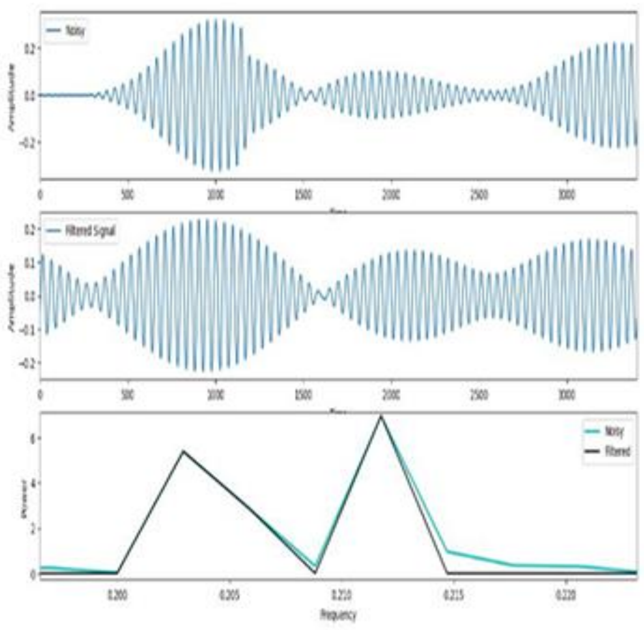


Fig.3. Filtering signal using FFT

Figure 3 illustrates how the Hilbert transformer is advantageous for extracting the envelope. This is explained by the formula that states the Hilbert transform yields a sin(t) for each cos(t) as HT provides a ±90° phase shift to the input signal. HT can produce the time signal's envelope by determining the time function's magnitude. The gradual overall change and sound intensity over time are analysed using envelopes.

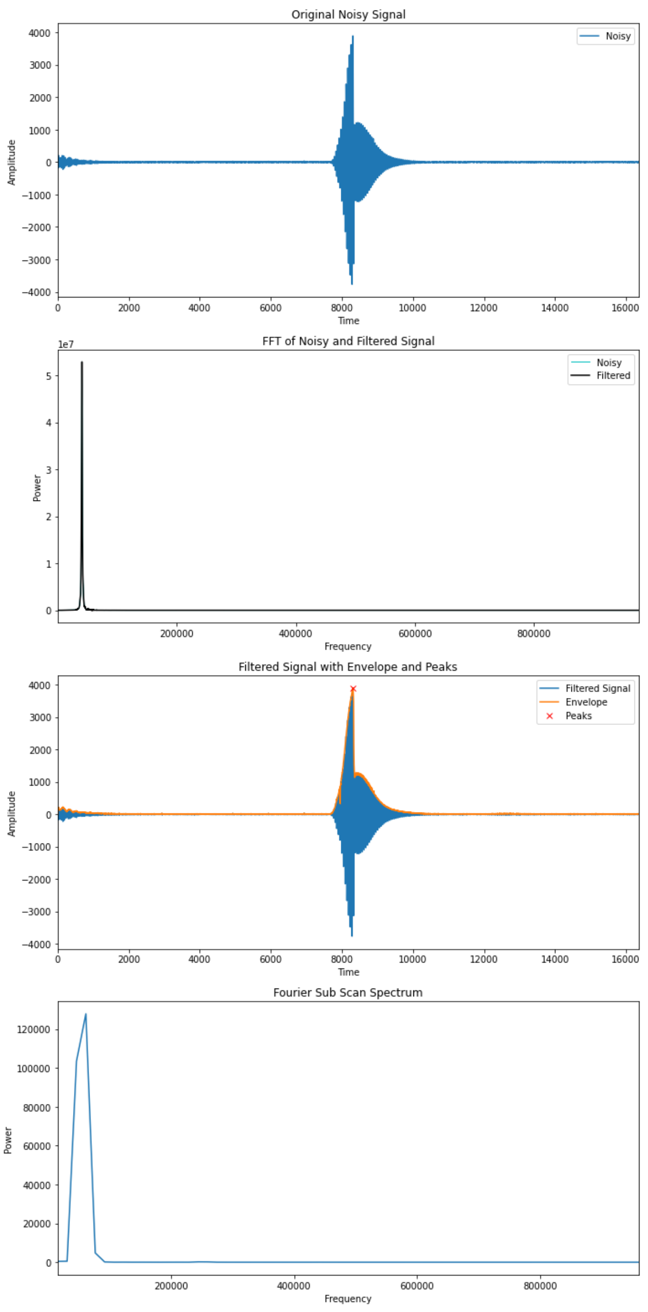


Fig.4. Performing FFT and Hilbert Transform for static object

Since the real signal may be analytically extended from the real axis to the upper half of the complex plane, the Hilbert transform can be used to identify an imaginary function to a real valued signal. The outcome is as shown in figure 4.

- *ML method used for finding first reflection.*

Implementing a machine learning method for identifying the first reflection in an audio or acoustic environment involves several key steps. Below is a generalized approach that you can follow:

1. *Define the Problem*: Clearly define the problem and the objectives of identifying the first reflection. Understand the

characteristics of first reflections in the context of your application. [13] [14]

2. _Data Collection_: Gather a dataset that includes audio recordings with labelled information about the presence and characteristics of first reflections. Ensure diversity in the dataset to capture various acoustic conditions. [13] [14]

3. _Data Preprocessing_:

- Audio Segmentation: Divide audio recordings into segments relevant to the analysis.
- Feature Extraction: Extract relevant features from the audio segments (e.g., spectral features, time-domain features, MFCCs).
- Labelling: Ensure accurate labelling of the dataset, marking segments with and without first reflections. [13] [14]

4. *Data Splitting*: Split the dataset into training, validation, and test sets to evaluate the model's performance [13] [14].

5. *Model Selection*: Choose an appropriate machine learning model based on the nature of the problem. Experiment with different algorithms such as SVM, decision trees, CNNs, or hybrid models depending on your dataset and problem requirements.

6. *Model Architecture*: Design the architecture of the chosen model, considering input features, hidden layers, and output layer. For deep learning models, experiment with architectures like CNNs or hybrid CNN-LSTM models [14].

7. *Feature Scaling and Normalization*: Standardize or normalize the input features to ensure that the model trains effectively and converges faster.

8. *Model Training*: Train the model using the training dataset. Fine-tune hyperparameters through iterative training and validation [13].

9. *Model Evaluation*: Evaluate the model's performance on the validation set to avoid overfitting. Adjust the model architecture or hyperparameters as needed [14].

10. *Testing*: Assess the final model on the test set to gauge its generalization performance.

11. *Post-Processing (if needed):* Implement any post-processing steps, such as filtering or thresholding, to refine the model's output.

12. *Interpretability and Visualization:* Analyse and visualize the model's predictions to gain insights into its decision-making process. Understand which features are most informative for detecting first reflections [14].

13. *Deployment*: If the model meets the desired performance, deploy it in the target environment. Consider real-time or batch processing depending on the application requirements [13].

14. *Continuous Monitoring and Improvement*: Implement a monitoring system to track the model's performance over time. Consider retraining the model periodically with new data to adapt to changing conditions. [13]

15. *Documentation*: Document the entire pipeline, including data preprocessing steps, model architecture, hyperparameters, and any insights gained during the development process.

4. _Machine Learning Model_:

As part of our project, we used two models:

1. Multi Layers Perceptron (MLP)
A multilayer perceptron (MLP) represents a type of feedforward neural network composed of fully connected neuron's employing nonlinear activation functions. The Multi-Layer Perceptron (MLP) is a type of artificial neural network that is used in this system to estimate distances from ultrasonic sensor data. It is well-suited for modelling complex relationships and patterns in the data, enabling more accurate detection of human presence.

1. *Data Preprocessing:* The dataset is pre-processed to ensure uniformity and suitability for input into the MLP model. This may involve tasks such as normalization, resizing, and augmentation to enhance the robustness and generalization capabilities of the model.

2. *Model Architecture:* The MLP consists of three main components: the **input layer**, **hidden layers**, and the **output layer**. The input layer receives features extracted from the sensor data, including time-domain and frequency-domain information. The hidden layers process this information using activation functions, and the output layer generates the final distance prediction.

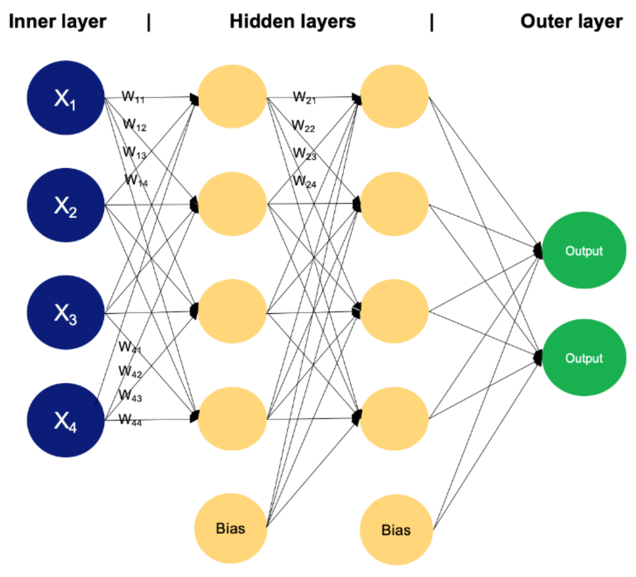


Fig.5. Architecture of MLP model

3. *Training:* Training the MLP involves feeding it sensor data and adjusting the network's weights to minimize prediction errors. The model is optimized using a loss function that measures the difference between the predicted and actual values. Preprocessing techniques, like signal

cleaning and feature extraction, are applied to ensure the input data is reliable.

4. *Hyper Parameters Tuning:* To optimize the MLP's performance, parameters like the number of layers, learning rate, and batch size are adjusted. This fine-tuning helps the model achieve better accuracy and efficiency.

5. *Model Evaluation:* The performance of the MLP is evaluated using error metrics, ensuring it accurately estimates distances. These predictions are essential for determining human presence and enabling the smart lighting system to function effectively.

2. Random Forest

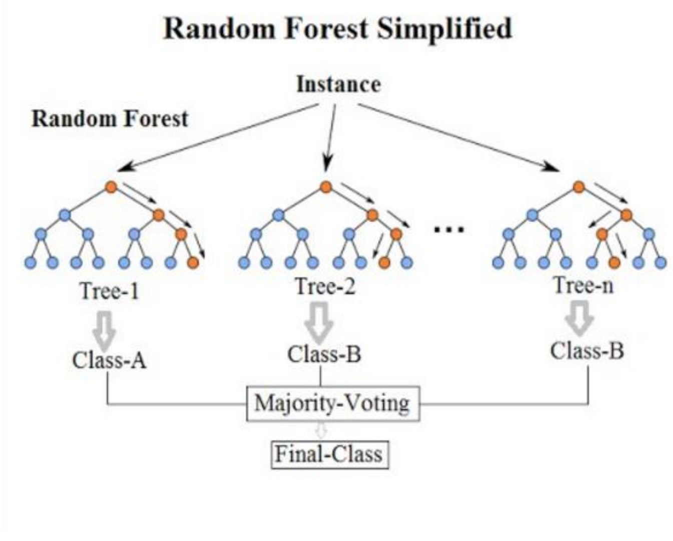


Fig.6.Random Forest

1. Data Preparation: The dataset is prepared by splitting it into training and testing sets to facilitate model training and evaluation.

2. Ensemble Learning: A random forest ensemble model is constructed by combining multiple decision trees. Each tree is trained on a random subset of the training data and features, introducing diversity and reducing overfitting.

3. Training: The random forest model is trained on the training dataset using the bagging (bootstrap aggregating) technique, where each tree is trained independently.

4. Feature Importance: The importance of each feature in the dataset is assessed based on its contribution to the overall performance of the random forest model. This information can help identify relevant features and improve model interpretability [15].

5. Evaluation: The trained random forest model is evaluated on the test dataset to measure its performance in terms of accuracy, precision, recall, and other relevant metrics [16].

5. *Confusion Matrix*:

A confusion matrix is a table that is used to define the performance of a classification or machine learning algorithm. By comparing the predicted labels of a collection of data to the actual labels, a confusion matrix table that summarises the performance of a classification method is obtained. An example of confusion matrix is shown in Fig. 7

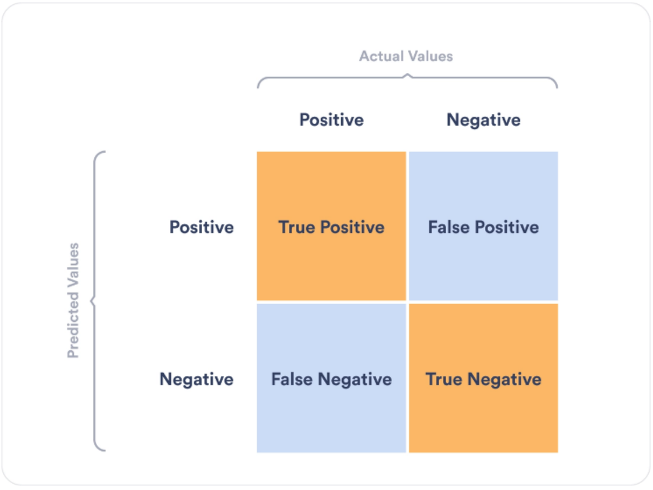


Fig.7.Confusion Matrix

The confusion matrix consists of four basic characteristics (numbers) that are used to define the measurement metrics of the classifier. These four numbers are:

1. TP (True Positive): When the expected value matches the actual value, we have a true positive case. This indicates that the model predicted a positive result, and the actual result is positive.

2. TN (True Negative): When the expected value and the real value are the same, a truly negative situation arises. This time, however, the model predicted a negative value, and the actual value is also negative.

3. FP (False Positive): When the expected value matches with an incorrect value, it known as a false positive. The model anticipated a favourable outcome even though the actual value was negative. This is called the type I prediction error.

4. FN (False Negative): When the expected value matches with an incorrect value, this is referred to as a false negative case. Even though the actual result was positive, the model predicted a negative result. This is the type II prediction error.

These four parameters are used to obtain other performance metrics which can be used to better judge the model.

Some additional performance metrics of an algorithm are accuracy, precision, recall, and F1 score, which are calculated on the basis of the above-stated TP, TN, FP, and FN[7].

Several significant metrics, including the following ones, can be computed using the confusion matrix to assess how well the classification algorithm performed:

Accuracy: The ratio of correctly predicted samples to all samples in the dataset is known as accuracy. It calculates the frequency with which the classifier predicts the right thing. The formula for accuracy is:

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

Precision: The ratio of accurately predicted positive samples to the total number of positive samples the model projected is known as precision. It counts the proportion of expectedly positive samples that really are positive. The precision formula is:

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

Recall (Sensitivity): The ratio of accurately predicted positive samples to all positive samples in the dataset is known as recall. It calculates the proportion of real positive samples that the model accurately predicted. The recall formula is as follows:

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

F1 Score: The harmonic mean of recall and precision is the F1 score. When recall and precision are both significant, it is a useful metric. The F1 score formula is:

$$\text{F1Score} = \frac{2*precision*recall}{precision+recall}$$

Recognising the widespread emphasis on accuracy when assessing performance is crucial before delving into confusion matrices, the foundation of model evaluation. Though it has its uses, precision by itself is not always reliable. Confusion matrices' applications and the drawbacks of accuracy as a stand-alone statistic will both be discussed in this section.

- True Positive Rate (TPR): Also known as recall, this metric represents the proportion of positive cases the model identified correctly (TP / (TP + FN)).
- False Positive Rate (FPR): This metric represents the proportion of negative cases the model incorrectly classified as positive (FP / (TN + FP))
- True Negative Rate (TNR): Also known as specificity, this metric represents the proportion of negative cases the model identified correctly (TN / (TN + FP)).
- False Negative Rate (FNR): This metric represents the proportion of positive cases the model incorrectly classified as negative (FN / (TP + FN)).

Depending on the issue and application, these rates can be helpful for assessing different parts of a classification model's performance. The model's ability to correctly identify positive cases is measured by TPR, while its inclined to incorrectly classify negative circumstances as positive is measured by FPR. The model's capacity to accurately identify negative scenarios is measured by TNR, while its inclined to incorrectly classify positive events as negative is measured by FNR. To create a high-performing classification model, we generally wish to maximise TPR and TNR while reducing FPR and FNR[8].

III. IMPLEMENTATION

A. *Measurement Environment and Setup*

Laboratory Configuration: The experiments were conducted in a controlled environment of the Machine Learning laboratory at the Frankfurt University of Applied Sciences. A controlled setting was essential to minimize external factors that could impact sensor performance and ensure reliable data collection.

Sensor and Object Placement: The FIUS sensor was mounted on a tall metal stand to maintain an unobstructed line of sight to the target object.The object/ person would be placed/stood below the sensor, was positioned to ensure measurement consistency throughout the experiment. This setup was designed to guarantee repeatable and accurate sensor readings.

Manual Verification Process: To calibrate the sensor and verify its accuracy, distance measurements were manually taken using a folding meter stick. This manual measurement process, crucial for validating sensor readings, is illustrated in Fig 6 and 7.

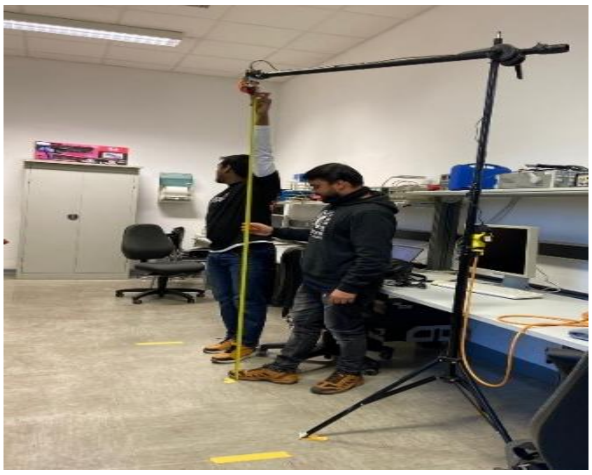


Fig.8. Manually measuring distance using a folding meter stick

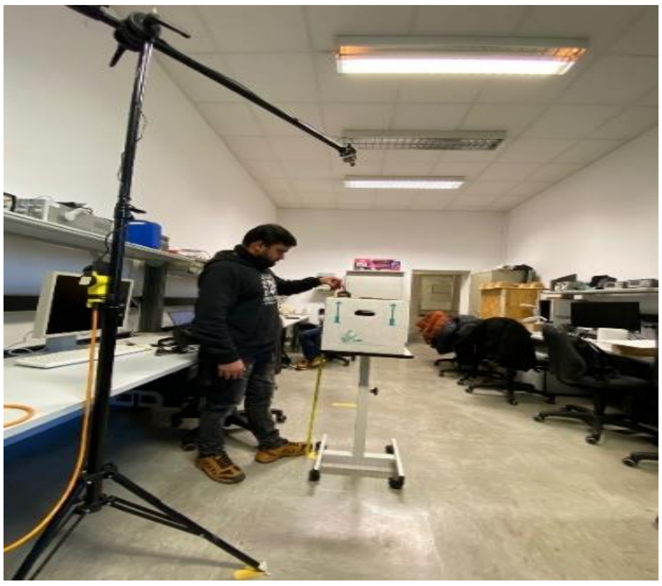
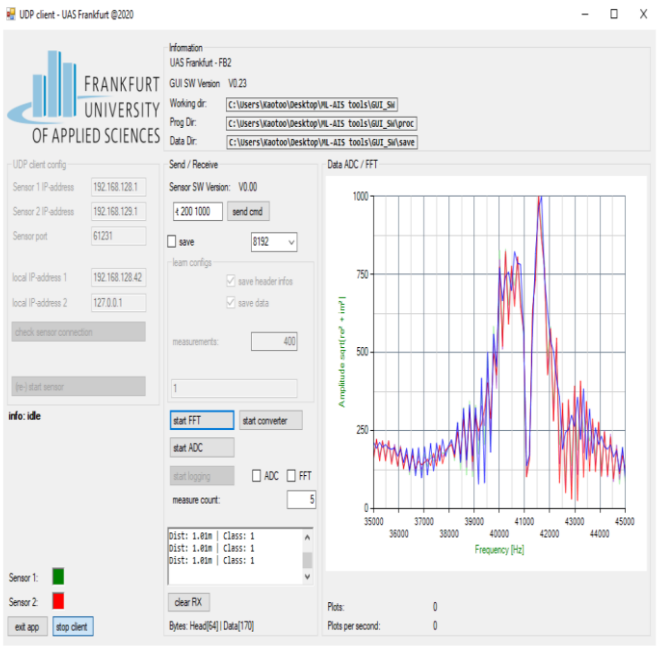


Fig.9. Manually measuring distance using a folding meter stick

B. *Measurement Software Utilization*

Software Interface and Functionality: The data from the Red Pitaya measurement board was collected using a custom measurement software developed in Frankfurt University of Applied Sciences. As shown in Fig.8, this software's graphical user interface (GUI) played a crucial role in the data collection process. It enabled users to perform real-time analysis and visualize sensor data, offering the flexibility to work with either raw analog-to-digitally converted signals or apply Fast Fourier Transform (FFT) for frequency domain analysis.

Data Acquisition and Analysis: The software facilitated the export of FFT-transformed data in a text format, making it ready for further processing and analysis. Additionally, it provided the ability to plot both FFT graphs and time-series dat, offering a detailed understanding of the sensor's performance under various testing conditions.


UDP client - UAS Frankfurt @2020

Information
UAS Frankfurt - FB2
GUI SW Version V0.23
Working dir: C:\Users\Kaotoo\Desktop\ML-AIS tools\GUI_SW
Prog Dir: C:\Users\Kaotoo\Desktop\ML-AIS tools\GUI_SW\proc
Data Dir: C:\Users\Kaotoo\Desktop\ML-AIS tools\GUI_SW\save

UDP client config
Sensor 1 IP-address: 192.168.128.1
Sensor 2 IP-address: 192.168.129.1
Sensor port: 61231
local IP-address 1: 192.168.128.42
local IP-address 2: 127.0.0.1

info: idle

Send / Receive
Sensor SW Version: V0.00
-t 200 1000
save: 8192
measurements: 400
measure count: 5

```
Dist: 1.01m | Class: 1
Dist: 1.01m | Class: 1
Dist: 1.01m | Class: 1
```

Bytes: Head[64] | Data[170]

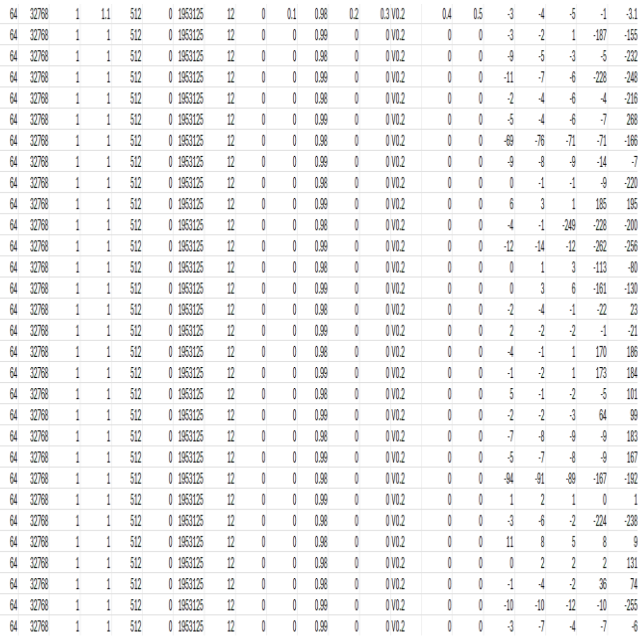
Plots: 0
Plots per second: 0

Fig.10. FFT data from the measurement software

C. *Data Structure and Export Format*

FFT Data and Headers: For this measurement, we used text-based ADC data that was exported from the software. Each data has 16384 columns overall, with rows representing amplitude values in the range of 0 to 1000 and columns representing frequency values.

Data headers are also exported by the program in addition to ADC data. The length of the data, the classification outcome from the software's current model, or the sampling frequency are just a few examples of the useful information included in data headers. Figure 9 shows a sample data format, and the table below, Table I, provides information on the data headers.



64	32768	1	1.1	512	0	1953125	12	0	0.1	0.98	0.2	0.3	V0.2	0.4	0.5	-3	-4	-5	-1	-3.1
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	-3	-2	1	-187	-155
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	-9	-5	-3	-5	-232
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	-11	-7	-6	-228	-248
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	-2	-4	-6	-4	-216
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	-5	-4	-6	-7	268
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	-69	-76	-71	-71	-166
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	-9	-8	-9	-14	-7
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	0	-1	-1	-9	-220
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	6	3	1	185	195
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	-4	-1	-249	-228	-200
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	-12	-14	-12	-262	-256
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	0	1	3	-113	-80
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	0	3	6	-161	-130
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	-2	-4	-1	-22	23
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	2	-2	-2	-1	-21
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	-4	-1	1	170	186
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	-1	-2	1	173	184
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	5	-1	-2	-5	101
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	-2	-2	-3	64	99
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	-7	-8	-9	-9	183
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	-5	-7	-8	-9	167
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	-94	-91	-89	-167	-192
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	1	2	1	0	1
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	-3	-6	-2	-224	-238
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	11	8	5	8	9
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	0	2	2	2	131
64	32768	1	1	512	0	1953125	12	0	0	0.98	0	0	V0.2	0	0	-1	-4	-2	36	74
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	-10	-10	-12	-10	-255
64	32768	1	1	512	0	1953125	12	0	0	0.99	0	0	V0.2	0	0	-3	-7	-4	-7	-6

Fig.11. Raw measurement data

TABLE.I DESCRIPTION OF THE HEADERS IN THE MEASUREMENT FILE

Column	Header Description	Value	Remarks
1	Header lengths	64	
2	Data length	170	
3	Class detected	1 or 2	1: Object 2: Human
4	Measurement type	0 or 1	0 : FFT 1 : ADC
5	Frequency resolution f or sampling time t depend on measurement type	119	
6	-	0	irrelevant
7	Sampling frequency (Hz)	1953125	
8	ADC resolution	12	
9		0.0	irrelevant
10	Distance between sensor and first object (round-trip-time in µs)	-	
11	FFT Window length	0	
12		0	irrelevant
13	software version (RP)	V0.2	

Data Collection:

After carefully setting up the measurement environment, we extracted ADC data as text files from the Red Pitaya board, which was connected to an echo-based ultrasonic sensor. This stage required precise calibration to establish baseline conditions for a thorough comparative analysis. Our goal was to create a dataset that reflected diverse real-world scenarios by varying factors such as distance, environmental conditions, and object materials. Stationary objects were placed at distances of 1 meter and 50 cm, and 1,000 Analog-to-Digital Converter (ADC) data points were systematically recorded for each object type.

In addition, we gathered extensive data for various human-related conditions, including walking, sitting, and standing scenarios. To ensure the robustness and reliability of the data, we repeated this process over two experimental runs. The large volume of collected data was essential for effectively training our model, enabling it to recognize and adapt to different real-world situations. These conditions were designed to mimic the challenges a smart light system would face in everyday use, ensuring its functionality in diverse environments.

- Detection of Static-Object:

In this phase, a rigid box was used as the static object to evaluate the smart light system's response to non-moving items. The box was selected for its reflective properties, which differ notably from those of organic materials like the human body. It was positioned at two specific distances—1 meter and 50 centimetres—from the ultrasonic sensor, mimicking the types of stationary objects commonly found in rooms, such as furniture. Precise measurements were taken using a standard measuring scale to ensure the accuracy of distance readings. Any discrepancies between the manual measurements and the sensor data were logged and analysed via the UDP client application. This analysis was crucial for refining the smart light system's ability to distinguish static objects, allowing the lights to turn off when no motion is detected, optimizing energy efficiency without misinterpreting reflections from non-moving items.

- Detection of Dynamic Objects:

To evaluate the smart light system's response to motion, dynamic objects such as a person walking, sitting, or standing were used in this phase. These scenarios mimic real-world conditions where movement should trigger the lights to stay on. The ultrasonic sensor was tasked with detecting motion from different distances and orientations. Data from the sensor was collected and analyzed to ensure the system accurately identified movement while filtering out minor, irrelevant motions. This detection of dynamic objects is critical for the smart light system, ensuring that the lights remain on only when human activity is present, thus preventing unnecessary energy consumption and enhancing user convenience.

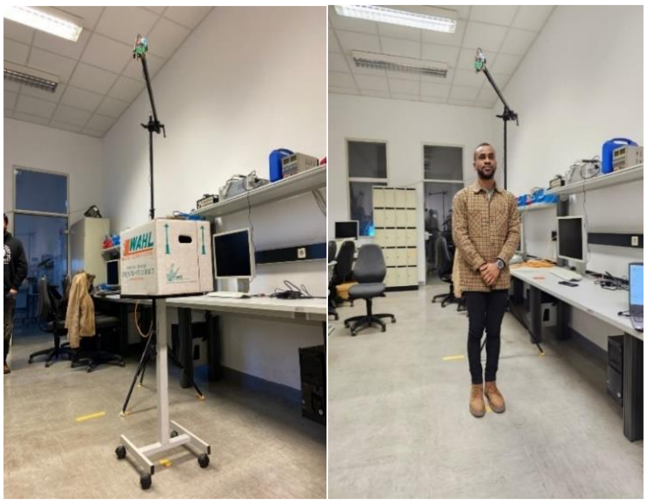


Fig.12. a. Detection of static object
Fig.12. b. Detection of dynamic object

D. *Data Preprocessing*

The implementation outlines a systematic approach to processing and analysing ultrasonic signal data for echo detection, following the steps detailed in the flowchart. The process begins with data collection, where we gather ADC data from ultrasonic sensors for various scenarios involving both static and dynamic objects. Next, we move to preprocessing, which includes signal windowing, noise reduction using techniques like the Hilbert transform, and data labelling. Each of these steps is crucial for enhancing signal quality and extracting relevant features. Following preprocessing, we perform feature extraction, identifying key time-domain and frequency-domain characteristics that are essential for effective model training.

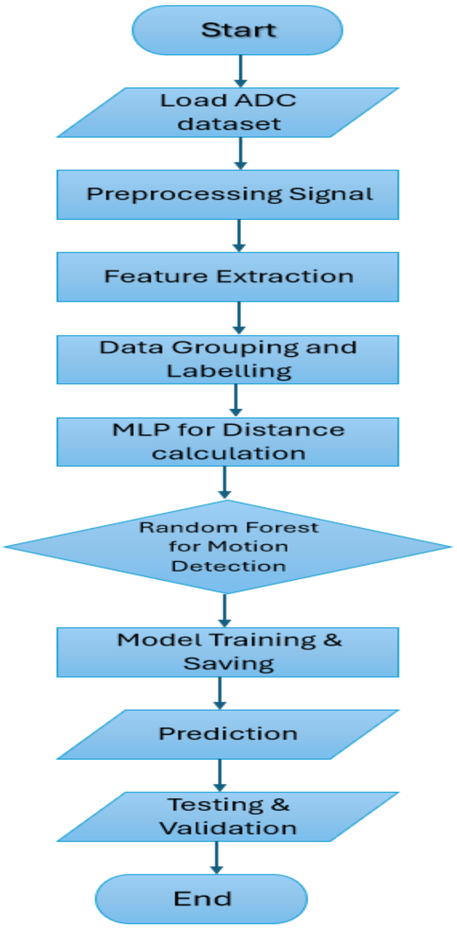


Fig.13. Flowchart of the Model

This meticulous preparation ensures that our machine learning models are trained on high-quality, meaningful data, maximizing their accuracy and effectiveness in detecting the first echo in ultrasonic signals.Fig.14 illustrates the flowchart of the methodology used to develop the model for detecting the first echo. The flowchart outlines each phase, including data collection, preprocessing, feature extraction, model training, and prediction, providing a clear visual representation of the entire process employed in the project.

Data Pre-Processing and Feature Extraction Steps:

Data Preprocessing: The initial phase involves reading and preparing the data for subsequent analysis. This step is crucial as it sets the foundation for the model's accuracy and reliability. The process begins with importing signal data from a CSV file. Given that not all columns in the file may be relevant to the analysis, a selection is made to focus on specific columns containing the essential signal data. This refined dataset is then subjected to further preprocessing to ensure it is in a suitable format for analysis.

Signal Windowing: Signal windowing is a pivotal preprocessing step designed to mitigate spectral leakage, a common issue in signal processing where energy from the signal 'leaks' into adjacent frequencies. To address this, a windowing function, specifically a Hanning window, is applied to each signal. The Hanning window, characterized by its smooth, sinusoidal shape, effectively tapers the signal at its beginning and end, thereby reducing spectral leakage. This process not only enhances the quality of the frequency analysis but also improves the model's ability to accurately identify echoes within the signal.

Noise Reduction and Features Extraction: The next step involves transforming each signal from the time domain to the frequency domain using Fast Fourier Transform (FFT). This transformation enables us to analyse the signal's frequency components, which facilitates the identification and removal of noise through a Power Spectral Density (PSD) thresholding method. Noise components with a PSD below a specified threshold are filtered out, and the cleaned signal is subsequently reconstructed in the time domain using the inverse FFT.

Following the noise reduction process, we extract the signal's envelope using the Hilbert Transform. The envelope, which represents the signal's amplitude modulation, is essential for detecting peaks within the signal. Peak detection is a crucial aspect of this process, as the locations of these peaks correspond to potential echoes in the ultrasonic signal. Identifying these peaks allows us to extract significant features indicative of the presence and location of objects or structures being analysed by the ultrasonic system.

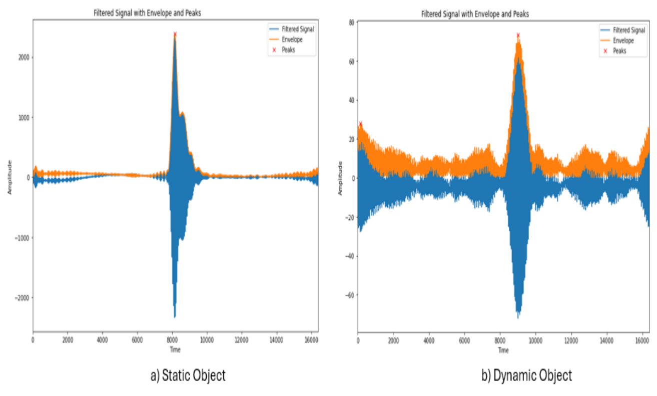


Fig.14. a. Signal Pre-Processing of static object
Fig.14. b. Signal Pre-processing of dynamic object

For feature extraction, we captured both frequency-based and time-based characteristics of the processed signal. The extracted features include magnitude of the first peak, RMS value, mean, max, variance, energy, frequency of the first peak, time of flight, PSD, and rate of change of distance. These features were then organized and stored in a separate CSV file, providing a structured dataset for further analysis and model training. This comprehensive feature set is essential for enhancing the accuracy of our machine learning models in detecting and classifying objects based on ultrasonic signals.

E. Labelling the data

In this experiment, we assigned labels to the data based on whether it represented a dynamic or static object, crucial for training machine learning models such as Random Forest and MLP. The labelling process involved categorizing data based on the nature of the detected object—either a static object (e.g., furniture or a stationary person) or a dynamic object (e.g., a moving person). The static objects were assigned a human label of 0, while dynamic objects were assigned a human label of 1. This binary classification is fundamental for distinguishing between scenarios where the smart light should stay off (static) or be triggered (dynamic).

During data collection, files were categorized into "human" (for dynamic objects) and "non-human" (for static objects). Based on this file naming convention, we automatically assigned the appropriate labels. For example, data collected from a person walking or sitting in motion was labelled with a human label of 1, indicating movement. On the other hand, non-human objects or humans in a stationary position were labelled with human label of 0, indicating no significant motion or static objects.

These labels serve as the target variable for our machine learning models. In the preprocessing stage, we extracted time-domain and frequency-domain features such as First Peak Magnitude, RMS Value, Mean, Max, Variance, Energy, First Peak Frequency, rate of change of distance and Time of Flight. These features, along with the corresponding labels, were stored in a CSV file. This structured dataset is then used to train and test our models

Ultimately, the data labelling process ensures that the machine learning algorithms can effectively differentiate between static and dynamic objects. This differentiation is crucial for the smart light system to optimize energy usage by turning off the lights when no significant motion is detected and turning them on when dynamic motion is present.

F. Training the Model

In machine learning, model training is the process where an algorithm learns from data to make accurate predictions. The goal is to teach the model how to identify patterns within the data and apply those insights to new, unseen scenarios. For our smart lighting system, the task involves detecting human presence and distinguishing between dynamic (moving) and static (stationary) objects based on ultrasonic sensor data. The data is pre-processed, and features are extracted, which are then used to train machine learning models.

To provide the best solution for our problem statement, two machine learning models were used. Using both the models together provided an efficient combination for solving the two key challenges in our system: calculating distances accurately and classifying objects to optimize smart lighting controls.

1. Multilayer Perceptron (MLP)
The Multilayer Perceptron (MLP) is a type of feedforward artificial neural network that is highly effective for regression tasks, where the goal is to model the relationship between input features and continuous output variables.

The MLP is well-suited for calculating distance and analysing time-based patterns in the data, such as the "rate of change of distance" between readings. Since our system relies on accurate distance measurements from ultrasonic sensors, the MLP's ability to learn non-linear relationships in the data makes it ideal for handling this complexity. By learning how distances vary over time, MLP helps the system detect moving objects in real time.

a) Model Initialization and Configuration
Model initialization and configuration involve setting up the MLP (Multi-Layer Perceptron) to be used for predicting human presence based on ultrasonic sensor data. This includes defining the structure of the model, such as the number of layers and neurons in each layer, which is crucial for capturing the complex patterns in the data. In our project, we utilize MLPRegressor from the sklearn library, configuring it to support early stopping to avoid overfitting during training. This step sets the foundation for the learning process, ensuring that the model is prepared to learn effectively from the input features.

b) Model Architecture
The architecture of the MLP consists of an input layer, one or more hidden layers, and an output layer. Each layer comprises a specified number of neurons, which are responsible for processing the input data. In our smart lighting system, the input layer accepts features extracted from ultrasonic sensor data, such as frequency magnitudes and time of flight. The hidden layers apply nonlinear transformations to the inputs, enabling the model to learn intricate relationships in the data. Finally, the output layer produces predictions of the accurate distance measurement and indicating the likelihood of human presence, which is essential for determining whether to turn the smart lights on or off.

c) Hyperparameters Tuning
Hyperparameter tuning is the process of optimizing the model's parameters to enhance its performance. In our implementation, we utilize GridSearchCV to explore various configurations of hyperparameters, including the number of hidden layers, the activation function (such as ReLU or tanh), the regularization strength (alpha), and the number of iterations (max_iter). By evaluating multiple combinations of these parameters through cross-validation, we can identify the configuration that yields the best predictive accuracy. This tuning process is critical for achieving optimal model performance and ensuring reliable predictions for the smart lighting system.

d) Model Compilation
Model compilation involves specifying the loss function and the optimization algorithm used to update the model weights during training. While the MLPRegressor does not require explicit compilation like deep learning frameworks, it internally uses a loss function that minimizes the mean squared error (MSE) during training. This loss function is appropriate for regression tasks, as it quantifies the difference between predicted and actual outputs. The optimization algorithm adjusts the weights of the model based on this loss, leading to better predictions over time.

e) Model Training
Model training is the process of fitting the MLP to the training data, allowing it to learn the underlying patterns and relationships. During training, the model iteratively updates its weights based on the input features and corresponding labels, using backpropagation to minimize the loss. Our implementation splits the dataset into training and testing subsets, ensuring the model learns from a diverse set of examples while maintaining a separate portion for evaluation. The training phase continues until convergence is reached, or early stopping conditions are met, which helps prevent overfitting.

f) Model Evaluation
Model evaluation is essential to assess the performance of the trained MLP. We employ cross-validation to obtain a robust estimate of the model's predictive capabilities across different subsets of the data. Key performance metrics, such as Mean Squared Error (MSE), R^2 score, precision, recall, and F1 score, are calculated to evaluate how well the model performs in predicting accurate distance measurement and human presence. This evaluation is crucial for understanding the model's effectiveness and ensuring that it can reliably classify human presence in the context of smart lighting.

f) Model Saving
After evaluating the model's performance and confirming its suitability for the application, we proceed to save the trained model for future use. This step is critical for deploying the smart lighting solution in real-world environments, allowing us to make predictions on new data without retraining the model. By saving the model, we ensure that the knowledge acquired during training can be utilized efficiently, facilitating the seamless operation of the smart lighting system in response to detected human presence.

Through careful implementation of the MLP, including initialization, configuration, tuning, and evaluation, we have developed a robust solution that enhances the accuracy of distance measurement and efficiency of smart lighting systems by accurately predicting human presence based on ultrasonic sensor data by some extend.

2. Random Forest

Random Forest is an ensemble machine learning method renowned for its high accuracy, flexibility, and ease of use. It constructs multiple decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forests are robust against overfitting, especially in cases where the dataset is large.

This model is perfect for classification tasks, such as differentiating between static and dynamic objects. Random Forest uses an ensemble of decision trees, making it highly robust to noise and capable of handling a wide range of features, like “First Peak Magnitude”, “RMS Value”, and Energy. Since our smart light system needs to classify whether the detected object is moving (dynamic) or stationary (static), Random Forest is particularly effective because it can handle both categorical and continuous features with high accuracy.

a) Hyperparameter Grid Setup

Before training the model, a comprehensive grid of potential hyperparameters is established to guide the search for the optimal model configuration. To ensure optimal model performance, a hyperparameter grid was created. Key hyperparameters include:

- n_estimators: The number of trees in the forest. Values considered are 50, 100, and 150, to balance between computational efficiency and model accuracy.
- max_features: The maximum number of features considered for splitting at each leaf node, with options 'sqrt' and 'log2' to test different feature subsets.
- max_depth: The maximum depth of each tree, including None (trees grow until all leaves are pure or contain less than min_samples_split samples), 10, 20, and 30, allowing for flexibility in tree complexity.
- min_samples_split: The minimum number of samples required to split an internal node, with options 2, 5, and 10, to prevent overfitting.
- min_samples_leaf: The minimum number of samples required to be at a leaf node, with values 1, 2, and 4, influencing the decision-making at the leaf level.
- bootstrap: Indicates whether bootstrap samples are used for building trees, with options True (bootstrap sampling) and False (the entire dataset is used to build each tree), affecting sampling variability.

b) Grid Search for Hyperparameter Tuning

Hyperparameter tuning is critical to optimize the model's predictive ability. GridSearchCV was employed, which tests different combinations of the hyperparameters in the grid using cross-validation. This ensures that the model's performance is consistent across different splits of the data and helps find the best possible configuration of parameters. The cross-validation process ensures that the model is not overfitting to a particular train-test split, providing a more generalized solution.

c) Thresholding

Thresholding plays a critical role in determining the performance of a classification model, especially in scenarios like the smart lighting system, where we need to optimize the decision to turn lights on or off based on sensor data. The Random Forest Classifier outputs probabilities indicating the likelihood of each class (human presence or absence). To convert these probabilities into binary predictions (e.g., "Human Present" or "Human Absent"), we set a decision threshold.

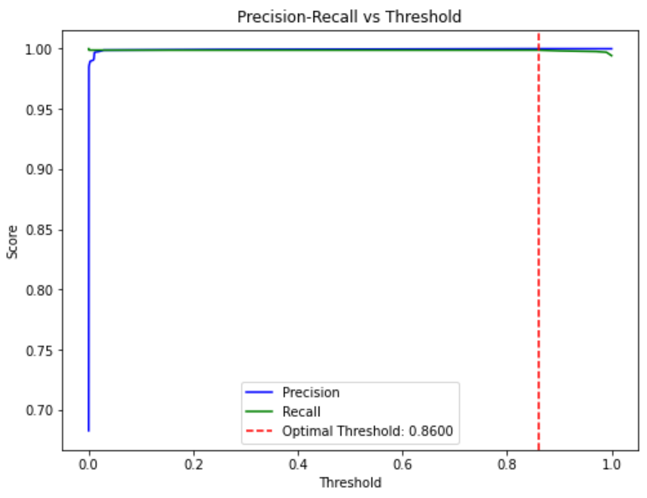


Fig.15. Precision-Recall Curve with Optimal Threshold for Random Forest Classifier

The plot illustrates the relationship between precision and recall across different threshold values. In this graph:

- Precision (blue line) measures how many of the predicted positives (human present) are actually correct.
- Recall (green line) measures how many actual positives (humans present) are correctly identified by the model.

As the threshold increases, the model becomes more conservative about predicting human presence, which improves precision but often lowers recall. Conversely, lowering the threshold increases recall but may introduce false positives, reducing precision.

The optimal threshold is where we find the best trade-off between precision and recall, which in this case is approximately 0.8600 (as marked by the red dashed line). This threshold ensures that the model makes reliable predictions without being too lenient or too strict, improving the system's balance between correctly turning the lights on when needed and avoiding unnecessary power consumption when no human is present.

By selecting this threshold, the smart lighting system can achieve high accuracy in differentiating between humans and static objects or background noise, optimizing both energy savings and user convenience.

d) Model Training with Optimal Parameters
Training involves feeding the Random Forest model the scaled dataset split into training and testing sets. During this phase, each decision tree in the forest is trained on a random subset of the data, and the trees collectively vote on the classification. This ensemble approach ensures that individual trees, which might have been overfitting, are balanced out by the other trees. For the smart lighting system, this training phase ensures that the model learns to distinguish between human presence and static objects effectively.

d) Model Evaluation
After training, the model's performance is evaluated on the test set using metrics such as:

- Accuracy: Measures the percentage of correct predictions.
- Confusion Matrix: Provides detailed insight into true positives, false positives, true negatives, and false negatives.
- Precision, Recall, F1 Score: These metrics further illustrate how well the model distinguishes between human and non-human objects, particularly in handling imbalanced classes.

Visualizing these metrics ensures that we can fine-tune the model if needed and detect any weaknesses in classification, such as a bias toward false negatives.

e) Model Saving
The best-performing model is serialized and saved as a pickle file to the file system. This allows the model to be persisted for future use without the need for retraining.

IV. RESULT AND ANALYSIS

A. *Signal Processing Characteristics for Static and Dynamic Objects*

Our experimental outcomes have uncovered significant signal processing distinctions between static and dynamic objects. With static objects, the signals demonstrate smooth, Gaussian-like envelopes indicative of singular, robust reflections typical of rigid, reflective materials. Specifically, the data for static objects display pronounced peaks in the time domain. Their Fourier spectra feature dominant frequency components, suggesting minimal dispersion of the ultrasonic waves.

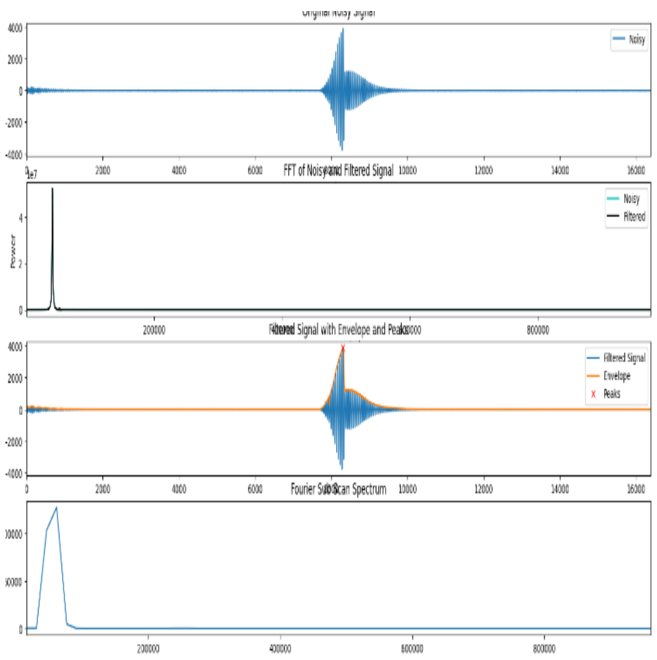


Fig.16. ADC to FFT plot for object placed at 1m.

In contrast, dynamic objects, such as a person standing, walking or sitting, generate signals with more complex spectral profiles, characterized by multiple peaks and a less defined envelope. This arises due to the ultrasonic waves scattering upon interaction with the varied shapes, leading to a broader spectrum with multiple frequency components.

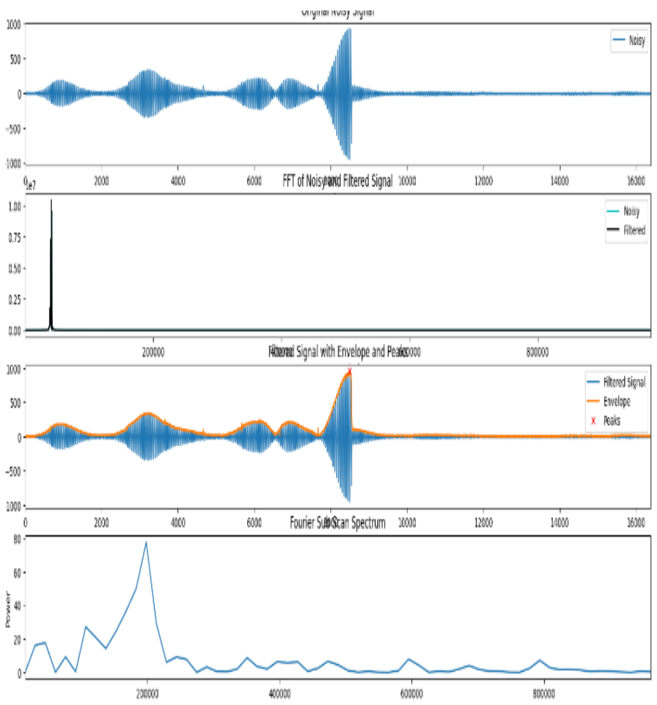


Fig.17. ADC to FFT plot for dynamic object.

Fig.18. ADC to FFT plot for a dynamic object (walking person) demonstrate the irregular envelopes and spectral plots with multiple peaks, indicative of the diverse echo signatures from moving targets.

These distinctive reflection signatures are critical for enhancing the sensor system's accuracy in discerning between object types.

B. *Model Performance for Object Classification*

The following sections provide a detailed analysis of the performance of the MLP (Multilayer Perceptron) and Random Forest models for detecting static and dynamic objects in a smart lighting system. The objective is to classify human presence and absence accurately to optimize energy usage by intelligently controlling the lighting system. The results of confusion matrices and performance metrics for various scenarios, such as static and dynamic object detection, are analysed to evaluate each model's effectiveness in achieving this goal.

1. Multilayer Perceptron (MLP)

The MLP model is evaluated across both static and dynamic environments. The model attempts to learn patterns from features extracted from the sensor data, such as distances, first peak magnitude and frequency, rate at which changes in distance occur and signal magnitude, to detect human presence or motion.

1. Static Object Detection

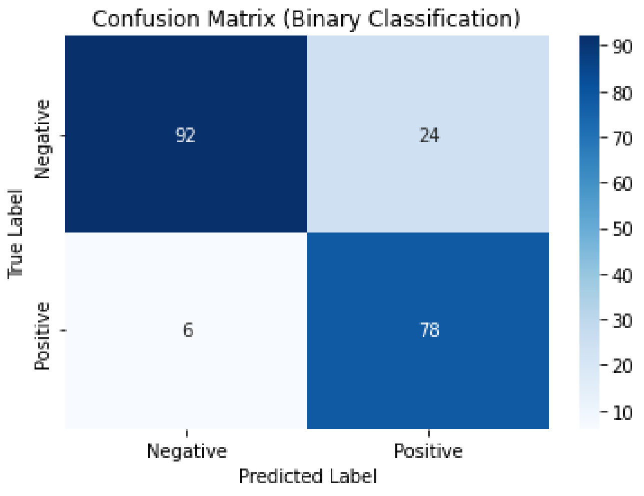
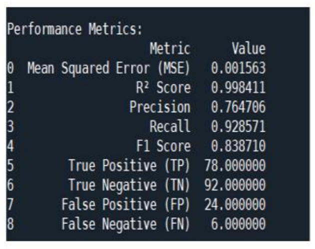


Fig.18. Confusion Matrix for a Static Environment.



```
Performance Metrics:
                   Metric      Value
0  Mean Squared Error (MSE)   0.001563
1                R² Score   0.998411
2               Precision   0.764706
3                  Recall   0.928571
4                F1 Score   0.838710
5      True Positive (TP)  78.000000
6      True Negative (TN)  92.000000
7     False Positive (FP)  24.000000
8     False Negative (FN)   6.000000
```

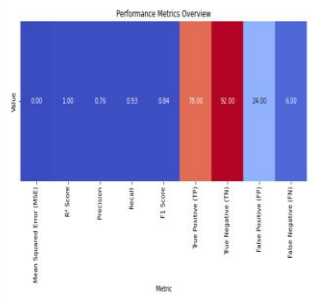


Fig.19. Performance Metrics for MLP for a Static Environment.

In this scenario, the environment is assumed to be static, meaning that no dynamic objects or people are present. The MLP model performs well, as reflected by a relatively high recall of 0.928571 and precision of 0.764706 (Refer Fig.19). The recall indicates that the model is highly effective in identifying true positives (i.e., correctly detecting no motion when no human is present), while the lower precision suggests that the model produces some false positives (misclassifying static objects as human presence).

This static environment serves as a baseline, representing a room where no significant motion occurs. The model's output is expected to be consistent with the absence of human activity. The high F1 score (0.838710) indicates a good balance between precision and recall, but the presence of false positives (FP = 24) suggests that the model occasionally misinterprets the static environment as dynamic.

The Mean Squared Error (MSE) is low, showing that the model's predictions closely match the actual data. The R² score of 0.998411 further confirms the model's strong performance, meaning that nearly 99.84% of the variance in the data is explained by the model.

2. Dynamic Object Detection

This section examines the model’s performance across different types of human movement, such as walking, sitting, and standing still. These scenarios represent dynamic environments where human presence needs to be accurately detected.

A) Person Walking/Moving:

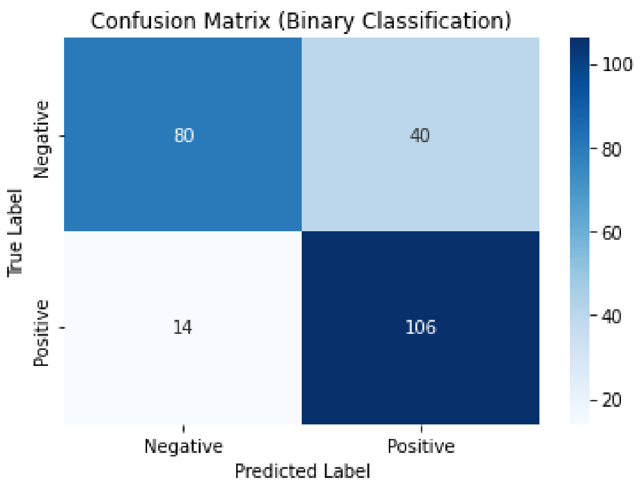
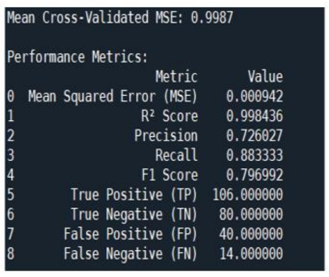


Fig.20. Confusion Matrix for person Moving for MLP Model



```
Mean Cross-Validated MSE: 0.9987

Performance Metrics:
                   Metric       Value
0  Mean Squared Error (MSE)    0.000942
1                R² Score    0.998436
2               Precision    0.726027
3                  Recall    0.883333
4                F1 Score    0.796992
5      True Positive (TP)  106.000000
6      True Negative (TN)   80.000000
7     False Positive (FP)   40.000000
8     False Negative (FN)   14.000000
```

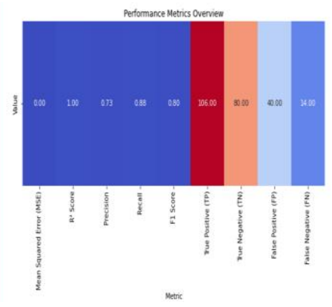


Fig.21. Performance Metrics for person Moving for MLP Model

For detecting a person walking, the model exhibits good recall (0.883333), indicating that it successfully detects the majority of dynamic instances. However, precision drops to 0.726027, suggesting that the model produces a noticeable number of false positives (FP = 40). This means the MLP model sometimes incorrectly identifies non-human activities as human movement, such as subtle environmental changes or background noise picked up by the sensors.

The F1 score of 0.796992 reflects a reasonable trade-off between precision and recall. The low MSE (0.000942) and high R² score (0.998436) confirm that the model fits the data well, but the lower precision points to room for improvement in correctly identifying human movement without false alarms.

B) Person Sitting:

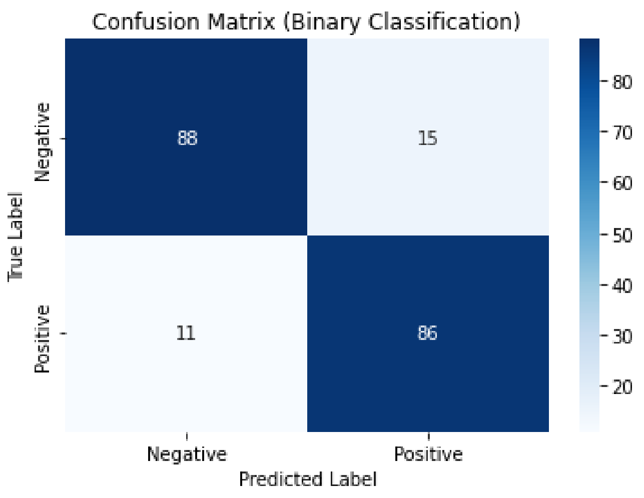
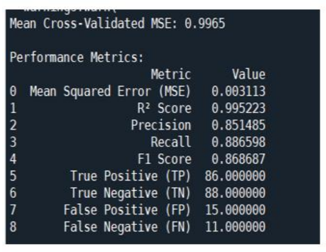


Fig.22. Confusion Matrix for person Sitting for MLP Model



```
Mean Cross-Validated MSE: 0.9965

Performance Metrics:
                   Metric      Value
0  Mean Squared Error (MSE)   0.003113
1                R² Score   0.995223
2               Precision   0.851485
3                  Recall   0.886598
4                F1 Score   0.868687
5      True Positive (TP)  86.000000
6      True Negative (TN)  88.000000
7     False Positive (FP)  15.000000
8     False Negative (FN)  11.000000
```

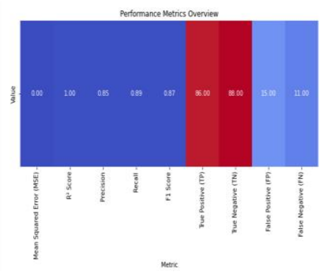


Fig.23. Performance Metric for person Sitting for MLP Model

The model performs well when detecting a person sitting, with high precision (0.851485) and recall (0.886598). The lower number of false positives (FP = 15) suggests that the model is better at distinguishing between human presence and absence compared to the "person walking" scenario. The F1 score of 0.868687 highlights a good balance between precision and recall, showing that the model is effective in identifying human presence even when movement is minimal (such as sitting).

The slightly higher MSE (0.003113) indicates that this scenario is more challenging for the model to predict compared to a person walking. However, the R² score of 0.995223 still shows a strong fit between the model's predictions and the actual data.

C) Person Steady:

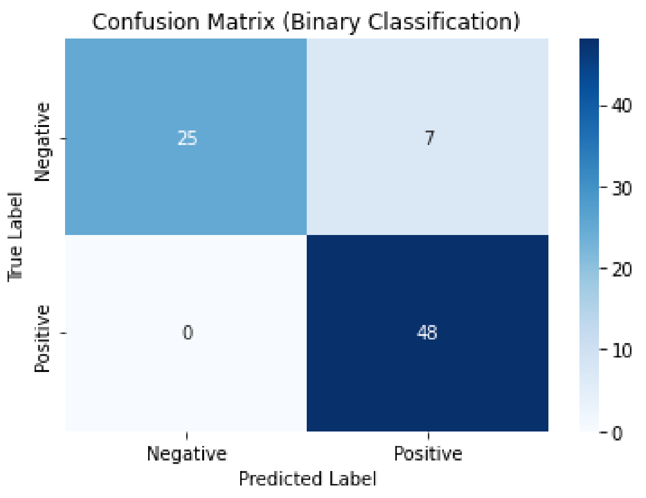
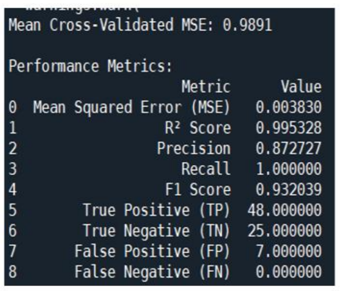


Fig.24. Confusion Matrix for person Steady for MLP Model



```
Mean Cross-Validated MSE: 0.9891

Performance Metrics:
                   Metric      Value
0  Mean Squared Error (MSE)   0.003830
1                R² Score   0.995328
2               Precision   0.872727
3                  Recall   1.000000
4                F1 Score   0.932039
5      True Positive (TP)  48.000000
6      True Negative (TN)  25.000000
7     False Positive (FP)   7.000000
8     False Negative (FN)   0.000000
```

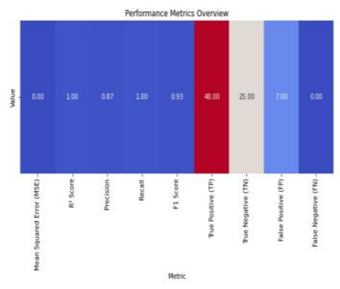


Fig.25. Performance Metric for person steady for MLP Model

In this scenario, the model excels, with perfect recall (1.0) meaning that it detects every instance of a person standing still without missing any. The precision of 0.872727 indicates that the model produces few false positives (FP = 7), making it highly effective in distinguishing between static and dynamic scenarios.

The F1 score of 0.932039 further demonstrates the model's strength in balancing precision and recall. The slightly higher MSE (0.003830) and an R² score of 0.995328 suggest that this scenario is more complex to model, but the performance is still excellent.

2. Random Forest

In this implementation, the model was evaluated using five-fold cross-validation, which involves dividing the dataset into five equal parts. Each of these parts serves as a test set once while the remaining four parts are used for training the model, effectively creating multiple training scenarios. This approach results in the model being fitted a total of 120 times, as it was trained across 24 different candidate scenarios with each scenario being assessed in all five folds. The aggregation of results from these decision trees enhances the model's ability to accurately classify static and dynamic objects based on the extracted features from the sensor data, leading to improved predictive performance and robustness in real-world applications.

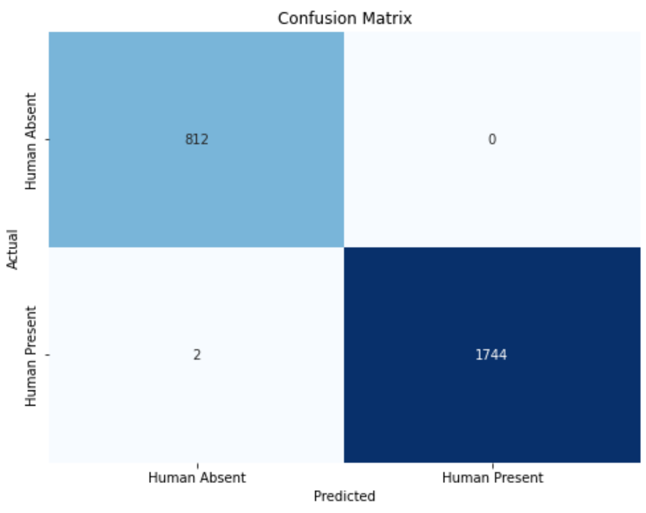
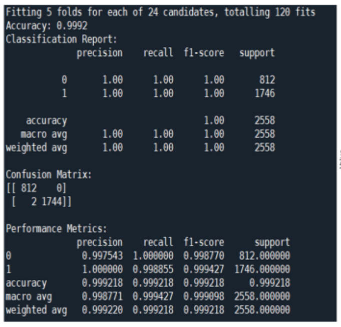


Fig.26. Confusion Matrix for random forest model



```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Accuracy: 0.9992
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       812
           1       1.00      1.00      1.00      1746

    accuracy                           1.00      2558
   macro avg       1.00      1.00      1.00      2558
weighted avg       1.00      1.00      1.00      2558

Confusion Matrix:
[[ 812    0]
 [   2 1744]]

Performance Metrics:
              precision    recall  f1-score      support
0              0.997543  1.000000  0.998770   812.000000
1              1.000000  0.998855  0.999427  1746.000000
accuracy       0.999218  0.999218  0.999218     0.999218
macro avg      0.998771  0.999427  0.999098  2558.000000
weighted avg   0.999220  0.999218  0.999218  2558.000000
```

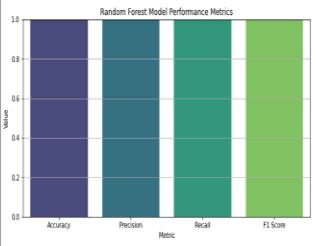


Fig.27. Performance Metric for person Moving for MLP Model

The Random Forest model demonstrates near-perfect performance in classifying static objects. With a precision of 0.997543 and a perfect recall of 1.00, the model almost never produces false positives or false negatives. The F1 score of 0.998770 indicates an ideal balance between precision and recall, making the model highly reliable in detecting the absence of human presence. The support value (812) indicates that a substantial number of data points were correctly classified as static.

For dynamic objects, the Random Forest model again delivers near-perfect results. The model has a precision of 1.00 and recall of 0.998855, meaning it almost always correctly identifies dynamic objects and rarely misses any human movement. The F1 score of 0.999427 indicates that the model handles both precision and recall exceptionally well, effectively detecting all dynamic movements. The support value (1746) shows that a large number of dynamic instances were correctly classified.

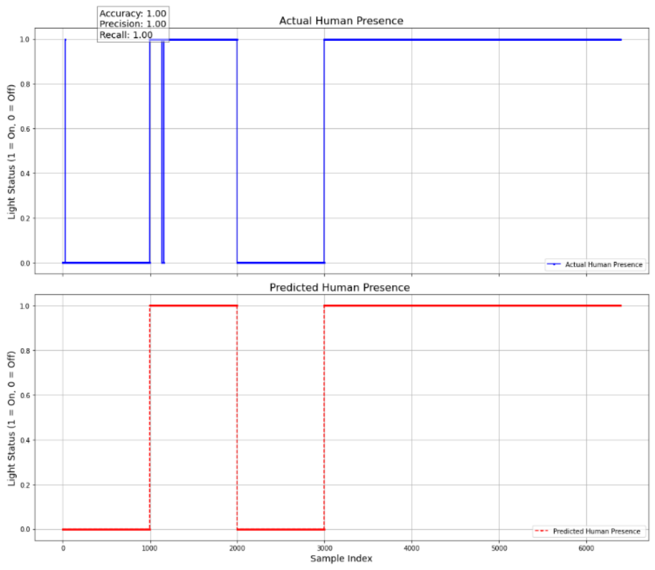


Fig.28. Visualization graph for prediction made by random forest model on unknown data.

The above plot shows that trained random forest model was tested on 7000 unknown samples. This model's overall accuracy is nearly perfect at 99.92%, reflecting its ability to generalize across different scenarios, whether static or dynamic. The macro average and weighted average F1 scores, both near 1.0, indicate that the model performs uniformly well across all classes.

The Random Forest model shows almost flawless performance across all scenarios, achieving perfect precision and recall in most cases. The model is highly robust and generalizes well to different environmental conditions, making it ideal for detecting both static and dynamic objects in real time. Its near-perfect accuracy and F1 scores suggest that it is the better candidate for smart lighting solutions, where accurate detection of human presence is crucial to optimize energy usage.

However, the MLP model performs reasonably well across all scenarios, especially in detecting a person standing still, where it shows perfect recall and a high F1 score. However, the model shows some limitations in precision when detecting more dynamic activities, like walking, where it produces more false positives. Overall, the MLP model achieves a good trade-off between precision and recall and proves effective in detecting human presence in both static and dynamic environments, but there is room for improvement in reducing false alarms.

V. CONCLUSION & FUTURE SCOPE

The objective of this project is to identify the presence of humans from ultrasonic waves to enhance smart lighting solutions. We utilized the Red Pitaya STEMLAB measurement board and an Ultrasonic sensor SRF02 for this purpose. Our findings revealed that ultrasonic waves behave differently when interacting with various materials, as evidenced by distinct patterns in the reflected signals captured by the sensor. The Red Pitaya board was instrumental in data collection, allowing us to clearly discern these patterns.

Additionally, we found that the classification behaviour observed in our experiments could vary significantly, especially when there were notable changes in the experimental setup. Through comparative analysis, we determined that each model has unique strengths in processing the data, contributing to the effective detection of human presence for our smart lighting system.

In the comparative analysis of the Multilayer Perceptron (MLP) and Random Forest models for detecting human presence and absence, several critical distinctions emerge that highlight their respective capabilities and limitations in the context of smart lighting systems.

The MLP model demonstrates satisfactory performance, particularly in static scenarios where minimal movement occurs, such as when a person is standing still. In these conditions, the model effectively identifies human presence, achieving a commendable balance between precision and recall. However, the MLP model exhibits some challenges in dynamic scenarios, where it tends to produce a higher number of false positives. This limitation arises from the model's tendency to misclassify rapid movements or changes in the environment, leading to occasional inaccuracies in detecting human presence. Despite these challenges, the MLP's capacity to discern subtle variations in environmental conditions positions it as a viable candidate for smart lighting systems. Nevertheless, further tuning and optimization of the model's parameters could enhance its precision and reduce misclassifications, particularly in environments with more dynamic movement.

In stark contrast, the Random Forest model consistently outperforms the MLP across nearly all performance metrics. It achieves near-perfect precision, recall, and F1 scores in both static and dynamic environments, indicating its robust ability to generalize from training data to unseen instances. The strength of the Random Forest model lies in its ensemble approach, where multiple decision trees are trained on various subsets of the data. This diversity allows the model to aggregate predictions and mitigate the risk of overfitting, making it highly reliable for detecting human presence accurately. With an impressive accuracy rate of 99.92%, the Random Forest model provides dependable and consistent performance, which is paramount for smart lighting solutions that rely on precise detection of human activity to optimize energy efficiency.

The findings of this analysis underscore the importance of selecting the appropriate model based on the specific requirements of the application. While the MLP model can serve effectively in controlled environments with minimal movement, the Random Forest model stands out as the superior choice for smart lighting applications that necessitate adaptability to varying human activities. Its high accuracy and ability to minimize false positives ensure that energy consumption is optimized, making it an excellent fit for systems aimed at reducing unnecessary energy use while maintaining a responsive environment for occupants.

In conclusion, although the MLP model is a competent tool for detecting human presence, the Random Forest model's superior performance and reliability make it the preferred option for smart lighting solutions. By effectively adapting to the dynamic nature of human behaviour, the Random Forest model not only enhances user experience but also contributes to energy conservation efforts, aligning with the broader goals of intelligent lighting systems. Further research and implementation of this model can pave the way for more sophisticated and efficient smart lighting technologies, ultimately improving the quality of indoor environments.

Future Scope:

The exploration of machine learning models for detecting human presence in smart lighting systems opens several avenues for enhancement and refinement. This section outlines potential improvements for both the Multilayer Perceptron (MLP) and Random Forest models to optimize their performance in various scenarios, particularly in detecting minimal motion.

Enhancements to the MLP Model: To bolster the MLP model's effectiveness, especially in dynamic scenarios, several strategies can be considered:

1. Hyperparameter Tuning: Fine-tuning hyperparameters such as learning rate, number of hidden layers, and neuron counts can significantly improve the MLP's ability to learn from data. Employing techniques like Grid Search or Random Search can help identify the optimal settings for the model.

2. Data Augmentation: Expanding the training dataset through data augmentation techniques can enhance the model's robustness. This may involve generating synthetic data that simulates various human activities and environmental conditions, thereby providing a more diverse set of training examples.

3. Feature Engineering: Introducing additional features or refining existing ones can improve the MLP's performance. For instance, incorporating temporal features that capture changes in distance over time could help the model distinguish between human presence and background noise more effectively.

4. Regularization Techniques: Implementing regularization methods, such as dropout or L2 regularization, can prevent overfitting. This is particularly important in scenarios with limited training data, ensuring that the model generalizes well to new, unseen situations.

5. Ensemble Methods: Combining the MLP with other models or using ensemble techniques can enhance its predictive capability. For instance, blending predictions from multiple neural networks can lead to improved accuracy and robustness.

Improvements to the Random Forest Model:

While the Random Forest model already demonstrates exceptional performance, further refinements can enhance its ability to detect minimal motion, such as when a human remains still or steady:

1. Advanced Feature Selection: Implementing more sophisticated feature selection techniques can help identify the most informative features relevant to detecting subtle changes in human presence. Utilizing techniques like Recursive Feature Elimination (RFE) or tree-based feature importance can streamline the feature set, leading to better model performance.

2. Class Weight Adjustment: Adjusting the class weights during training can enhance the model's sensitivity to less frequent classes, such as 'human present' in minimal motion scenarios. This adjustment encourages the model to prioritize accurate detection of humans even when their movement is negligible.

3. Threshold Optimization: Fine-tuning the classification threshold based on the distribution of predicted probabilities can improve the model's ability to differentiate between human presence and absence. This is particularly critical in scenarios where the distinction is subtle.

4. Time-Series Analysis: Incorporating time-series data can enhance the model's understanding of patterns associated with human presence. Analyzing the temporal dimension of the data can allow the model to learn from sequential information, aiding in the identification of humans who may be motionless but still present.

5. Integration of Sensor Fusion: Combining data from multiple sensors can improve the accuracy of detecting human presence. For example, integrating visual data from cameras or additional environmental sensors (like temperature or light) can provide complementary information, enhancing the Random Forest model's decision-making process.

6. Ongoing Learning: Implementing online learning techniques that allow the model to adapt and improve with new incoming data can ensure continued relevance and accuracy. This approach is particularly useful in dynamic environments where human behaviour may change over time.

By focusing on these future enhancements, both the MLP and Random Forest models can be optimized for more accurate and reliable detection of human presence, even in the face of minimal motion. Such advancements will contribute significantly to the development of smarter, more efficient lighting systems that respond intelligently to occupant behaviour, ultimately enhancing user experience and promoting energy efficiency.

VI. ACKNOWLEDGMENT

I would like to express my gratitude to Prof. Dr. Andreas Pech for giving me the chance to work on this subject under their supervision and for giving me the right direction to finish the required project and this report. The lectures on "Machine Learning" were very beneficial in providing the background knowledge. If I do not express our gratitude to the writers of the references and other works cited in this work, we will be in breach of our responsibility.

VII. REFERENCES

[1] Richards, Mike, Pi Updates and Red Pitaya, Radio User. Bournemouth, UK: PW Publishing Ltd. 11, 2016.

[2] R. C. Luo, S. L. Lee, Y. C. Wen and C. H. Hsu, “Modular ROS Based Autonomous Mobile Industrial Robot System for Automated Intelligent Manufacturing Applications,” 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), [Online]. Available: doi: 10.1109/AIM43001.2020.9158800..

[3] “SRF02 Ultrasonic range finder,” robot-electronics, [Online]. Available: https://www.robot-electronics.co.uk/htm/srf02tech.htm.

[4] P. N. T. Wells, “Ultrasonic attenuation measurements in liquids and solids,” *Physical Acoustics,* vol. 3, pp. 219-294, 1966.

[5] S. G. Kim, “Ultrasonic reflection and refraction at material boundaries,” *Journal of the Acoustical Society of America,* vol. 39, pp. 600-607, 1966.

[6] M. P. B. M. A. R. J. H. Arnold, “Multipath effects in ultrasonic waveforms,” *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control,* vol. 38, pp. 314-323, 1991.

[7] A. H. a. N. P. M. a. M. R. Pech, “A new Approach for Pedestrian Detection in Vehicles by Ultrasonic Signal Analysis,” pp. 1-5, 2019.

[8] Z. F. H. e. al., “A low-cost ultrasonic distance measurement system,” *IEEE* , 2013 .

[9] A. J. H. a. P. M. Grant, “Precision distance measurement using pulsed ultrasonics,” 1993.

[10] D. J. N. a. A. D. S. Fitt, “Distance measurement using ultrasound,” *Journal of Physics E: Scientific Instruments,,* 1982.

[11] K. F. W., Hilbert Transform, Vol. 2, Cambridge, UK: Cambridge University Press, 2009.

[12] B. R., The Fourier Transform and it`s application, McGraw-Hill, 1965.

[13] K. J. Piczak, “ENVIRONMENTAL SOUND CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS,” *IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP),* p. 6, 2015.

[14] Z. C. L. R. W. John R. Hershey, “Deep clustering: Discriminative embeddings for segmentation and separation,” 2015.

[15] L. Breiman, “Random forests,” *Machine Learning,* vol. 45, pp. 5-32, 2001.

[16] T. K. Ho, “Random decision forests,” in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, Montreal, QC, Canada, 1995.

[17] Y. B. a. G. H. Y. LeCun, “Deep learning,” *Nature,* vol. 521, pp. 436-444, 2015.

[18] I. S. G. E. H. A. Krizhevsky, “ImageNet classification with deep convolutional neural networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1,* Lake Tahoe, Nevada, 2012.

[19] T. C. a. C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016.

[20] A. L. a. M. Wiener, “Classification and regression by randomForest,” *R News,* vol. 2, pp. 18-22, 2002.

[21] P. S. a. M. J. E. Golay, “Smoothing and differentiation of data by simplified least squares procedures,” *Analytical Chemistry,* vol. 36, pp. 1627-1639, 1964.

[22] S. Mallat, “A theory for multiresolution signal decomposition: The wavelet representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 11, pp. 674-693, 1989.

[23] H. G. Mohamed-Elamir Mohamed, “A machine learning approach for ultrasonic noise”.

[24] S. T. B. M. G. C. C. R. C. L. M. R. D. J. Sohl-Dickstein, “A device for human ultrasonic echolocation,” 2015.

[25] Y. a. Y. S. Nagashima, “‘Ultrasonic sensing for a mobile robot to recognize an environment”.

[26] G. D. C. B. M. M. P. a. A. T. C. Canali, ““A temperature compensated ultrasonic sensor operating in air for distance and proximity measurements”.