Parameter setting and reliability test of a sensor system for person detection in a car wearing winter wear.

Master of Engineering
(M.ENG)
Information Technology
Aiman Zehra
1388996
aiman.-@stud.fra-uas.de

Master of Engineering
(M.ENG)
Information Technology
Alam Sher Khan
1387324
alam.khan@stud.fra-uas.de

Master of Engineering
(M.ENG)
Information Technology
Soundarya Talawai
1390133
soundarya.talawai@stud.fra-uas.de

Master of Engineering
(M.ENG)
Information Technology
Athkar Praveen Prajwal
1394663
praveen.athkar@stud.fra-uas.de

***Abstract*—Systems are frequently given the ability to learn from experience and improve themselves automatically without being particularly designed thanks to machine learning, which is sometimes referred to as the most popular new technology of the industrial revolution. Machine learning aims to find patterns and meaningful correlations on its own using examples and observations. Our major goal is to evaluate the performance of a Red Pitaya sensor system for human detection in an automobile when a person is dressed for winter. The sensor examines ultrasonic signals that the obstruction has backscattered. It consists of an ultrasonic transducer, an embedded system for managing the transducer and gathering signals, as well as a computer for processing signals and extracting features. Readings from the sensor are taken while taking into account different thresholds and use cases in order to train and later identify whether a human is spotted in the car seat. Random Forest classifier is used to complete the project, and a number of thresholds are taken into account when a person is dressed for winter in order to produce the best results and improve sensor detection. The outcomes are also assessed based on the F1 score generated by the confusion matrix.**

Keywords—F1 Score, Confusion matrix, Red Pitaya Sensor, Random Forest classifier.

I. INTRODUCTION

Today, a wide range of sensors are frequently employed in interior settings to enhance quality of life. A car's interior can be equipped with sensors, in particular, to detect a human sitting or a pet left alone in the vehicle. Non-contact passive sensors, such as camera, ultrasonic, and radar sensors, can also be used to monitor people. Unfortunately, a camera sensor's functionality declines in dimly light areas, and it can violate people's privacy. Using a Red Pitaya sensor system based on ultrasonic signals, prior systems' drawbacks have been overcome. It has substantially lower costs than other options like computer vision because it only requires an ultrasonic sensor and an embedded system.

Clear differences in the frequency spectra may be seen when comparing the signals that were backscattered from individuals and hard things (such as boxes, etc.). These variations might be influenced by the objects' surfaces. These findings imply that it may be able to differentiate between people and hard things using advanced ultrasonic signal processing.

Machine learning is a method that enables systems to automatically learn from experience and improve without being explicitly programmed. It is usually regarded as one of the most well-liked newest technologies in the fourth industrial revolution. Machine learning is said to be a subset of the artificial intelligence (AI) which has grown rapidly in recent years in the context of data analysis and computation, and frequently enables programs to function intelligently [1].

Since, AI systems handle the more fascinating tasks, like email spam detection, photo tagging, and web search. Hence, machine learning was created as a new capability for computers, and it now has an impact on numerous industries and fields of basic science. The effectiveness and efficiency of a machine learning solution are determined by the characteristics and nature of the data as well as the quality of the learning algorithms [2]. The focus of machine learning research is largely on selecting or creating an algorithm and running trials using that algorithm as the foundation. Such a skewed perspective lessens the impact of practical applications [3].

For machine learning to advance, learning must occur organically without human intervention or oversight. It is the condition of self-learning without overt programming. Recognizing the facts and trends is beneficial. The goal is to make it possible for computers to pick up new skills over time and adjust their behavior accordingly. It discusses the ineffective solutions to a variety of learning issues using a wide range of learning techniques. It provides strong support for psychological phenomena by practical investigations, theoretical analysis, or evaluation [4].

It is crucial to comprehend the fundamentals of different machine learning algorithms and how they can be used in a wide range of practical contexts, including IoT systems, cybersecurity services, business and recommendation systems, smart cities, healthcare, context-aware systems, sustainable agriculture, and many more [2]. In AI is not about flawlessly imitating humans; rather, it is about understanding the rules that enable agents to act intelligently and outperforming us. The fact is that intellect is no longer a property of solely humans. Machine learning has not only given rise to the idea of autonomous computing by making machines autonomous, but it has also lessened the constant watchfulness users must have over the applications [3].

Despite significant progress, there are still apparent flaws in the data set that machine learning is based on. Although learning is a continual process, it can be fixed by consistently keeping the data sets current. Considering all these drawbacks, machine learning has already resolved several significant global issues. Data mining, artificial intelligence, optical character recognition (OCR), statistics, computer vision, mathematical optimization, and other areas have all shown that machine learning is incredibly helpful, and its significance is only expected to grow. Because there are countless applications for machine learning, it is still an active area of research with vast potential for advancement and a bright future [3].

Considering Autonomous Intelligence system, the development of technology makes it possible to reach several milestones easily and precisely. Indeed, autonomous systems are crucial to both the industry and daily lives for people. More advanced artificial intelligence technologies are included into these systems to boost their capacity [5].

In this research, we will train a model to assess whether a human can be spotted in the car seat using readings from a Red Pitaya sensor, based on the significance and potential of "Machine Learning."

The following is a list of this paper's main contributions:

- Our objective is to test the dependability of a sensor system for human detection in an automobile while wearing winter clothing by adjusting various parameters.
- Random Forest classifier is developed and trained at various thresholds to obtain the ideal output using F1 score of confusion matrix.

II. THEORETICAL BACKGROUND

A summary of the project's various theoretical components will be provided in this part such as the operating hardware specifications, programming language used, and integrated underlying algorithm.

A. *Machine Learning*

For machine learning to advance, learning must occur organically without human intervention or oversight. The goal implements to make it possible for computers to pick up new skills over time and adjust their behavior accordingly. It discusses the ineffective solutions to a variety of learning issues using a wide range of learning techniques and provides strong support for psychological phenomena by pragmatist investigations, theoretical analysis, or evaluation [4].

The four primary categories of machine learning algorithms are reinforcement learning, unsupervised learning, semi-supervised learning, and supervised learning. Following, we quickly go through each sort of learning method and the extent to which it can be used to address difficulties in the actual world [2].

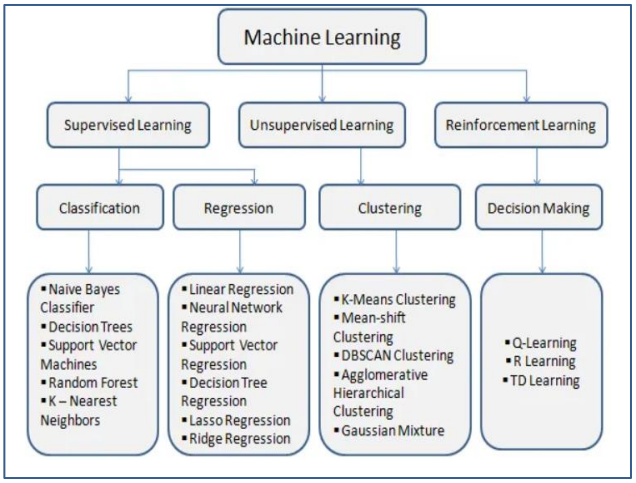


Figure 1: Types of Machine Learning [18]

i. *Supervised Machine Learning*

Supervised learning is utilized when there are input variables and target values for the output. The algorithm learns the mapping function from the input to the output. Given the availability of large scale labeled data samples, it is a costly method for applications when data is scarce. These methods can be broadly classified into two categories: Classification and Regression [6].

ii. *Unsupervised Machine Learning*

When there is no corresponding output variable and only the data in the form of an input, unsupervised learning is used. These algorithms simulate the underlying patterns in the data to gain a better understanding of its properties. Clustering is one of the key categories of unsupervised algorithms. This method uses the data's inherent groupings to find output predictions for unknown inputs. Predicting consumer purchase behavior would be an illustration of this strategy [6].

iii. *Reinforcement Machine Learning*

While making a series of decisions leading to a final reward, reinforcement learning is used. An artificial agent receives rewards or penalties for the activities it takes during the learning process. To maximize the overall reward is its aim. Examples include teaching robots to play video games or to carry out specific robotic duties [6].

B. *Performance evaluation using Confusion Matrix*

The accuracy of a Classification model is well explained by a confusion matrix. Normally, a confusion matrix will display a classifier's errors. A common categorization output format is a confusion matrix. It reveals the relationship between the projected category and the samples' innate categorization. The confusion matrix for an n-class classification issue is a $n \times n$ matrix. Information regarding the degree of polymerization within a category and the degree of dispersion between categories is also provided by the

confusion matrix [8].

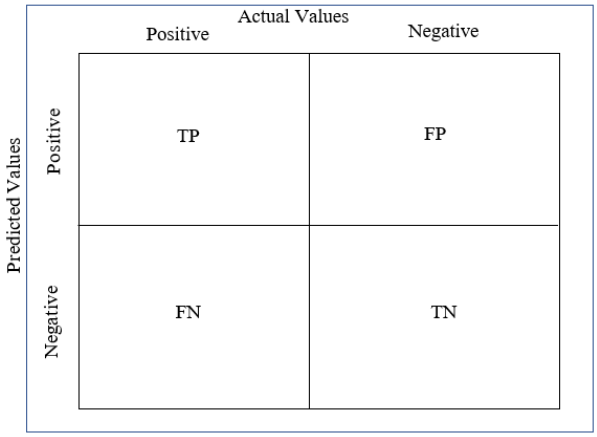


Figure 2: Confusion Matrix

Confusion Matrix lists the True Positive (TP) and True Negative (TN) in order to assess the classifier's accuracy. False Positive (FP) and False Negative (FN) results are also displayed, as seen in Figure 2. [7] The confusion matrix is a forecasting analytical technique found in machine learning. The effectiveness of a classification-based machine learning model is assessed using the confusion matrix. The number of accurate and wrong predictions made by a classifier (or a classification model) for binary classification tasks is summarized in a table known as the confusion matrix. The efficiency of a classification model is evaluated using a N x N matrix termed a confusion matrix, where N is the total number of target classes. By displaying the confusion matrix's total number of target classes, one might judge the model's accuracy by counting the diagonal values for accurate classifications. [9]

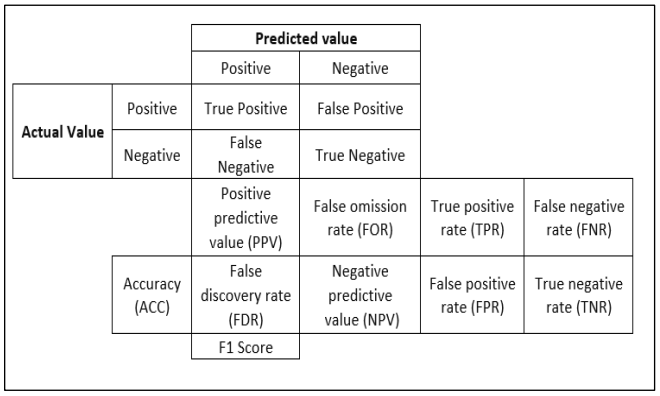


Figure 3: Confusion Matrix configuration and terminology

a. **True Positives (TP):** when both the projected value and the actual value are positive. [9]

b. **True Negative (TN):** when both the predicted value and the actual value are negative. [9]

c. **False positives (FP):** The situation where the prediction is positive, but the fact is negative. The is also known as Type 1 mistake. [9]

d. **False Negative:** when the fact is favorable, but the forecast is unfavorable. also referred to as the Type 2 mistake. [9]

e. **True Positive Rate (TPR), Sensitivity, Recall:** It is determined by calculating the ratio of the total number of Positive samples by the number of Positive samples accurately categorized as Positive. The true positive rate, or recall, is also known as sensitivity. A higher engagement score means the model is good at spotting positive examples. A lower recall score, on the other hand, implies that the model isn't very adept at detecting positive cases. [10]

$$\mathrm{TPR} = \frac{\mathrm{TP}}{(\mathrm{TP} + \mathrm{FN})} = \frac{\mathrm{TN}}{\mathrm{P}} \qquad (1)$$

f. **True Negative Rate (TNR), Specificity:** The classifier's specificity can be defined as the efficiency of the system to evaluate the false (negative) tests, it they are essentially false. [11]

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} = \frac{\mathrm{TN}}{\mathrm{N}} \qquad (2)$$

g. **False Positive Rate (FPR), fall-out:** It is the likelihood that someone who tests positive and does not have an illness. [12]

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} = \frac{\mathrm{FP}}{\mathrm{N}} \qquad (3)$$

h. **False Negative Rate (FNR), miss rate:** It is the possibility that someone who tests negative for an illness does. [12]

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}} = \frac{\mathrm{FN}}{\mathrm{P}} \qquad (4)$$

i. **Positive Predictive Value (PPV), Precision:** Measurement of precision is done by calculating the ratio between the number of correctly identified Positive samples to the total number of Positive samples. Precision can be thought of as a metric for how accurate or good something is. Mathematically, precision can be represented as:

$$\mathrm{PPV} = \frac{\mathrm{TP}}{(\mathrm{FP} + \mathrm{TP})} \qquad (5)$$

j. **Negative Predictive Value (NPV):** The likelihood that someone who tests negative for an illness does not actually have it.

$$\mathrm{NPV} = \frac{\mathrm{TN}}{(\mathrm{FN} + \mathrm{TN})} \qquad (6)$$

k. **False Omission Rate (FOR):** A performance measure called False Omission Rate shows the likelihood of a real positive in the event of a negative prediction. It is determined as:

$$\text{FOR} = \frac{\text{FN}}{(\text{FN} + \text{TN})} \qquad (7)$$

l. **False Discovery Rate (FDR):** A performance measure called False Omission Rate shows the likelihood of a real positive in the event of a negative prediction. It is determined as:

$$\text{FDR} = \frac{\text{FP}}{(\text{FP} + \text{TP})} \qquad (8)$$

m. **Accuracy:** Accuracy is a statistic that sums up how well a model performs. It is computed by the ratio of number of right guesses and the total number of forecasts. In mathematical form, accuracy is represented as:

$$\text{Accuracy Score} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FN} + \text{TN} + \text{FP})} \qquad (9)$$

n. **F1 Score:** The F1-score is a measurement of a test's accuracy in binary categorization analysis. It is indeed measured using the test's precision and recall, with precision equaling the true positives results divided by number of favorable findings, including those that were incorrectly identified, and recall equaling the number of true positive results divided by the total number of specimens that is identified as positive. [10]

$$\text{F1 score} = \ = \frac{(2\text{TP})}{(2\text{TP} + \text{FN} + \text{FP})} \qquad (10)$$

The F-score, also known as the F1-score, is a metric for determining how accurate a model is on a given dataset. It's used to assess binary classification algorithms that categorize examples as either 'positive' or 'negative.' The F-score, which is defined as the geometric mean of the model's accuracy, is a way to integrate the model's precision and recall. The F-score is a popular metric for evaluating information retrieval systems like browsers, as well as a variety of machine learning models, particularly in natural language processing [10].

C. Red Pitaya Sensor:

Red-Pitaya is a low-cost analog signal generating and measuring microchip STEM Lab board. Due to the Red Pitaya's twin fast ADC (Analog-to-Digital Converter) and dual fast DAC (Digital-to-Analog Converter), it is simple to simultaneously acquire and produce two sine waves with arbitrary phase differences. It runs Linux and may be controlled or accessed in a variety of ways, including browsers on a PC or tablet (it can run a web server), usb-serial console, and SSH protocol [13].

A computer with an ARM processor, an FPGA, and an electronic board with ADCs and DACs are all embedded into Red Pitaya board, which is a lab equipment replacement. [13] The Red Pitaya has an ultrasonic sensor installed on it that we utilized. The board serves as a platform for processing sensor signal data as well as classifying the signal after network algorithms training.

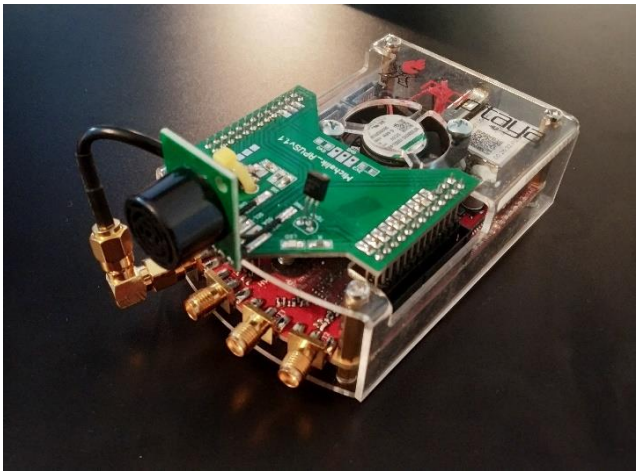


Figure 4: Red Pitaya with inbuilt ultrasonic sensors

D. Ultrasonic sensors:

An ultrasonic sensor is a piece of equipment that measures the distance to a target item using ultrasonic sound waves and then converts the sound's reflection into an electrical signal. Audible sound travels at a faster rate than ultrasonic waves do (i.e., the sound that humans can hear). An ultrasonic sensor primarily consists of the transmitter (which produces sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target) [14].

Most often, proximity sensors are combined with ultrasonic sensors. They are present in anti-collision safety systems and self-parking automotive technologies. Robotic obstacle detection systems and manufacturing technology both use ultrasonic sensors. Ultrasonic sensors in proximity sensing applications are less prone to interference from smoke, gas, and other airborne particles than infrared (IR) sensors are (though the physical components are still affected by variables such as heat) [14].

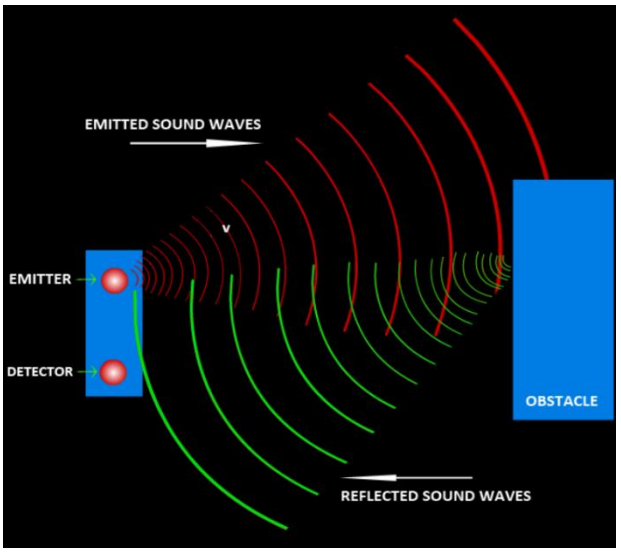


Figure 5: Ultrasonic Sensor [19]

E. Random Forest Classifier:

Random Forest is a general principle of classifier combination that employs L tree-structured base classifiers with N=1, 2, 3, and L, where X stands for the input data and n is a group of identical and dependent distributed random vectors. Random forest classifier has certain benefits such as: [15]

- Eliminating the issue of overfitting.
- They are less susceptible to outlier data in training data.
- Setting parameters is simple, which reduces the need for tree trimming.
- Automatically created variables with varied importance and accuracy.

The Random Forest is appropriate for large dimensional data modeling because it can allow missing values and handle continuous, categorical, and binary data. Because Random Forest is robust enough to manage overfitting difficulties due to the bootstrapping and ensemble approach, there is no need to prune the trees. In addition to having strong prediction accuracy, Random Forest is efficient, clear, and non-parametric for various dataset types. Random Forest provides a very unique combination of model interpretability and prediction accuracy when compared to other well-known machine learning techniques [15]. Because ensemble techniques and random sampling are used, precise predictions and superior generalizations are made. An undetermined collection of alternative trees with K random features at each node are combined to create a random tree. In this context, the term "at random" refers to a group of trees where each tree has an equal probability of being sampled. Alternatively, we could say that the distribution of trees is "uniform". Random trees may be produced effectively, and combining several such random trees typically results in models that are realistic. In the field of machine learning, there has been substantial research on random trees in recent years [15].

The strongest classification algorithm is the Random Forest Algorithm, which is based on the idea of a D-tree algorithm. The tree is built using the bootstrapping technique. The bootstrap method substitutes features and samples from the dataset with randomly chosen ones to produce a single tree. Like decision tree methods, random forest algorithms choose the best based on the feature splitters selected for categorization. The random forest algorithm also makes use of the Gini index and data [17]. To obtain access to resources for finding the best splitter. This process should continue until n trees are produced by the random forest. A forest with n decision trees can be made using the random forest algorithm. The problem of detecting accuracy is one that many trees address at a high level [17].

Figure 6 shows the representation of random forest. Random forests are a classification ensemble technique. Building decision trees using the provided training data and comparing them to the test data is part of the technique. The ranking of variables in a classification task is accomplished using random forests. We fit a random forest to the data to determine the relative relevance of each variable in the data set $D_n=(X_i,Y_i)i=1n$.

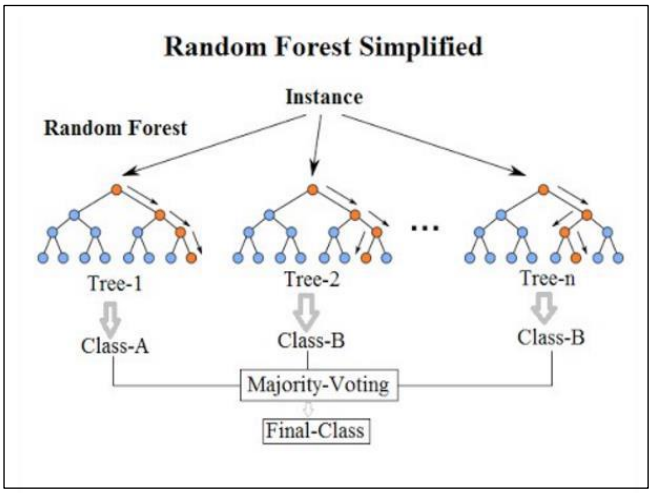


Figure 6: Random Forest [17]

The error for each data point is calculated throughout the fitting process and averaged over the forest. Characteristics that generate high values for this score are more significant than those that generate low ones. The proximity of the data points to one another and the significance of a variable are both revealed by random forests [16].

Following training, the i^{th} feature's values are permuted among the training data, and the error is once again computed on this set of data to assess the feature's importance. The relevance score for the i^{th} feature is calculated by summing the error difference between all the trees before and after the permutation. The score is normalized using the standard deviation of these variations [16].

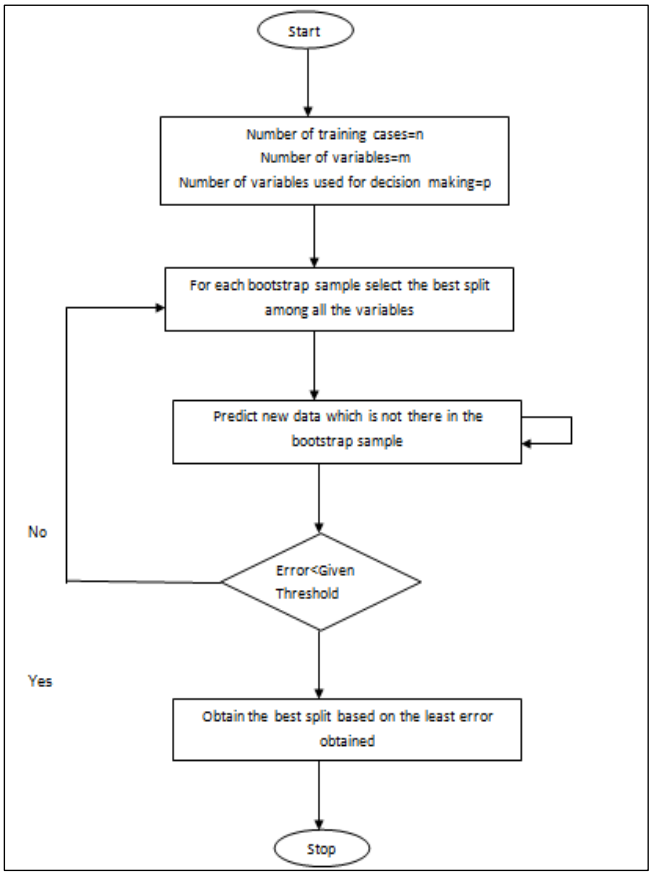


Figure 7: An illustration of the Random Forest method [16]

Disadvantages of Random Forest method is:

- Random forests are biased in favor of attributes with higher levels when categorical variables with varying levels are present in the data [16].
- Smaller groups are preferred over bigger groups if the data contain groupings of correlated features that are similar in relevance to the outcome [16].

The Random Forest Method is used in:

1. Is employed for pixel analysis in image categorization [16].
2. Is employed in the analysis of complex biological data in the field of bioinformatics [16].
3. Is employed for segmenting videos (high dimensional data) [16].

III. Methodology

The project methodological approach consists of 5 phases as mentioned below:

- Phase 1: Setup Hardware and Software
- Phase 2: Dataset Acquisition
- Phase 3: Creation of Confusion Matrix, F-1 score, accuracy and further observations.
- Phase 4: Develop and implement classification software for analysis of the already recorded data sets.
- Phase 5: Comparison of performance

A. *Setup Hardware and Software*

For setting up the hardware, in the car where the experiment is conducted, there are two red pitaya sensors installed. Depending on the placement of the sensor, a wireless connection is made to one of the sensors with the help of the GUI software. The linked sensor's IP address is 192.168.129.1. The Red Pitaya used in our project has been depicted below for measurement purposes in its true environment. It is fastened to the car's dashboard and used to record readings on the passenger side. Several use cases and variants have been tested.

Figure 8: Two Red Pitaya placement in the car actual environment

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

i. Once the UDP Client software is loaded and the wireless network credentials for the particular sensor (in our case Sensor 1) is provided then the green signal at bottom left indicates that connection to ‘Sensor 1’ is successfully established.

ii. The GUI saves the readings in a designated folder. The folder's or directory's full path is ‘\\GUI_V0.23_2021-11-22\GUI_SW\save’.

iii. The command which is used to set a particular threshold for taking the readings is ‘-t X Y’ where X and Y are the different threshold value combinations. The command is used to gather data, and it logs the readings at the value of X and Y. The command "-d 1" also offers demo mode capabilities, and the command "-d 0" turns off the demo mode. The ‘send cmd’ button is used after the thresholds have been selected to save the configuration, which can be modified for various values.

iv. The total number of readings that were recorded for every use case under every threshold is set in the 'measurements' block in which one measurement data set = 400 measurements per threshold and variation.

v. For each reading, the file name can be customized and changed in the blank column. It is shown as "P2ONBM" in this instance (the file naming convention used in this project has been explained in later sections).

vi. The 'FFT' checkbox must be selected before clicking on the ‘Start FFT’ button to record the readings. Once all the measurements are done then the GUI program automatically stops taking the measurements and the file is saved by the designated name.

The same steps are repeated when taking measurements for all use cases for all the considered thresholds.

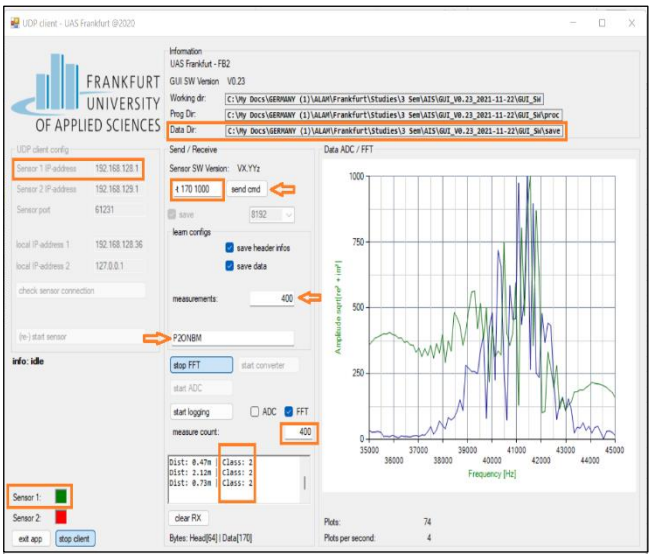


Figure 9: GUI used for taking measurements

B. *Acquisition of Dataset*

In this phase we collected raw data using the Red Pitaya

sensor for various use cases and thresholds. After analysis, a confusion matrix is generated, and conclusions are drawn. We collected a total of three datasets and the details, confusion matrix and observations from each of the dataset is described below:

1. *Dataset 1*

i. To record this dataset, a human passenger wearing winter wear is placed in the vehicle directly in front of the sensor (on the passenger side), with a predetermined distance separating the sensor and the human subject. The red pitaya Sensor 1 is set up in front of the passenger seat in horizontal position, as seen in the illustration below:

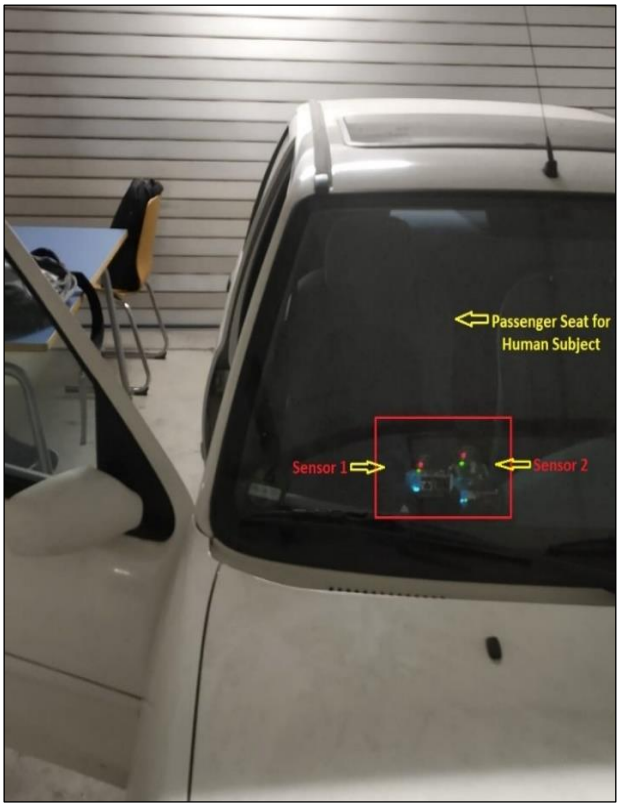
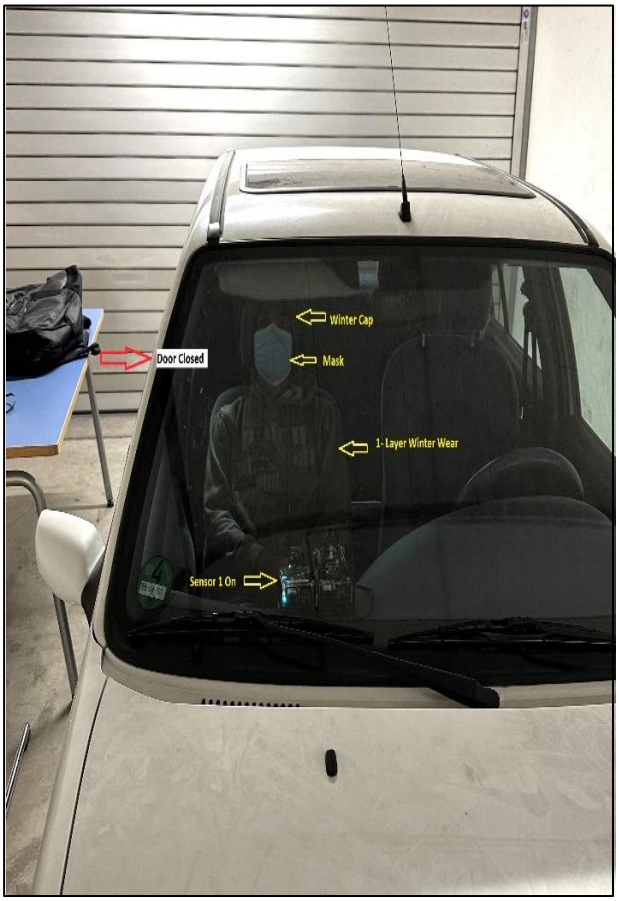


Figure 10: Red Pitaya placed in front of sensor seat

ii. For the 1st dataset, the default threshold was used to take the readings for all the use cases considered. In every use case, only one parameter was changed like the human subject is dressed in 1 layer of winter clothing, the effect of seatbelt, normal human motion in the car, open door or closed, etc. and the readings were taken to analyze the impact of that parameter on the sensor's classification efficiency. The below table provides the details used to create combinations for various use cases considered in the 1st Dataset and describes an example use case 'P6CNTS':

Table 1: Uses Cases for 1st Data Set

Variations in Clothing	Code
Multi-Layer Winter Wear	1
Multi-layer Winter Wear + Cap	2
Multi-layer Winter Wear + Cap + Mask	3
1-Layer Winter Wear	4
1 layer Winter Wear + Cap	5
1 layer Winter Wear + Cap + Mask	6
Variations in Arrangement	**Code**
Person / Empty Seat	P / E
Closed Door / Open Door	C / O
Seat Tilted / Not Tilted	T / NT
Passenger Moving / Still	M / S
Example Use Case	**File Name**
Person + 1 Layer Winter Wear + Cap + Mask + Closed Door + Seat Not Tilted + Still	P6CNTS



iii. From the above table, a total of 20 variations were used to create the use cases and the measurements were recorded for every human subject. Moreover, measurements were made while there was no human subject in the vehicle (use case: empty seat 'E'). The process is then repeated for all the different thresholds, and the observations are recorded for different human subjects.

iv. In the first dataset a total of 22800 readings were recorded and confusion matrix was created, and important observations were drawn about the classification efficiency of the sensor like accuracy, F1-score, etc.

Figure 11: Human test subject with multi-layer winter clothing, a cap and a mask.

v. Once the measurements for the 1st Dataset are completed, the classification files generated by the sensor using the GUI were preprocessed using a Python code (as mentioned in later sections) and resulting confusion matrix and the related performance matrices of the Red Pitaya sensor classifier are shown below:

		Predicted value			
		Positive	Negative		
Actual Value	Positive	True positive (TP) = 2462	False negative (FN) = 8938	True positive rate (TPR) = 21.59%	False negative rate (FNR) ≈ 78.40%
	Negative	False positive (FP) =171	True negative (TN) = 11229	False positive rate (FPR) = 1.5%	True negative rate (TNR) ≈98.5%
		Positive predictive value (PPV) ≈ 93.505%	False omission rate (FOR) ≈ 44.319%		
	Accuracy (ACC) =0.6004	False discovery rate (FDR) = 6.494%	Negative predictive value (NPV) = 55.68%		
		F1 score = 0.35088			

Class 1 is classified as Empty - **NEGATIVE**
Class 2 is classfied as Human - **POSITIVE**

Figure 12: Confusion Matrix for 1st Dataset (Red Pitaya Sensor Classifier)

After analysing the correct and incorrect predictions from the 1st Dataset readings and from the above confusion matrix the following observations were drawn:

- The use case with 1-Layer Winter Wear-Door Closed-Not Tilted-Movement shows the highest number of correct predictions.
- The cases with human subject with 'No Movements' shows higher number incorrect predictions (false negatives) from the sensor classifier.
- Movement outside the car near the passenger door created interference while readings were taken when the door was open.
- The resulting F1-score was 0.35088 which is lower than 0.5 due to a very high number of false negatives since the use cases with 'No Movement' gave high number of false negatives.

2. *Dataset 2*

i. For the 2nd Dataset, we collected a total of 64000 measurement readings. Similar to the dataset 1, for the 2nd dataset we followed the same human subjects, the same sensors but we explored some new use cases and tested the use cases with each of the new thresholds (X and Y) in order to get the values of X and Y at which we get better results. We considered observations from Dataset 1st to create new use cases for Dataset 2nd.

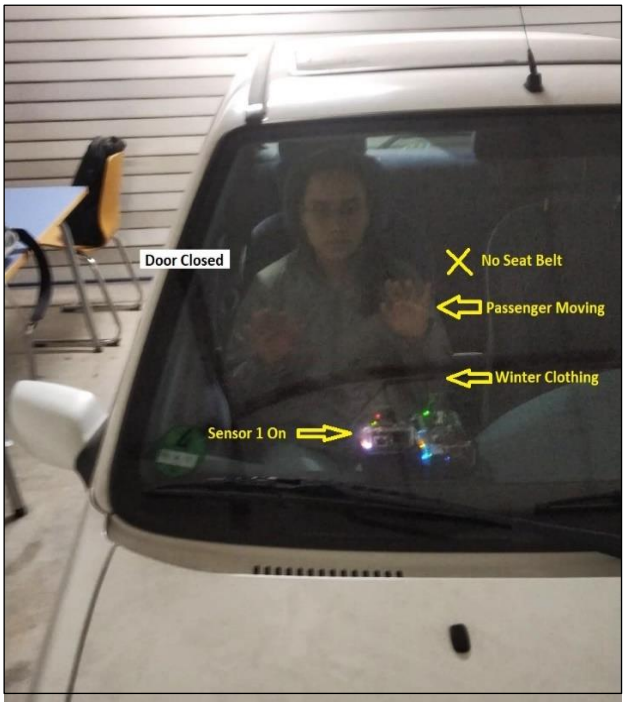
ii. For the threshold values of 'X' the values were varied in steps of 5 or 10 starting from 155 till 190 and for the 'Y' threshold we considered value of 1000 and 10000 for different values of 'X' as shown in the below table:

Table 2: X and Y threshold varying in predefined steps for the Dataset

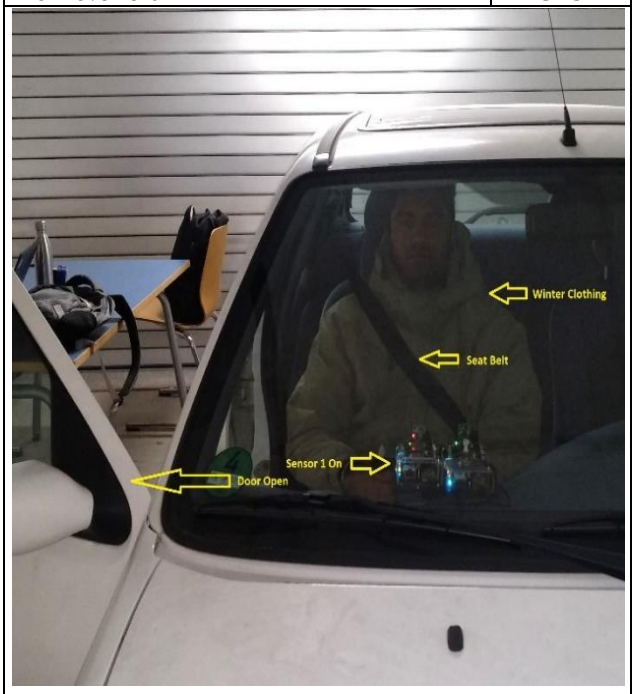
-t	X	Y
1.	160	1000
2.	170	1000
3.	180	1000
4.	185	1000
5.	190	1000
6.	155	10000
7.	165	10000
8.	175	10000
9.	180	10000
10.	185	10000

iii. The below table provides the details used to create combinations for various use cases considered in the 2nd Dataset and also describes two example use cases:

Use Cases	
1. Door Open – Winter Clothes – No Belt - Still	
2. Door Open – Winter Clothes – No Belt - Movement	
3. Door Closed – Winter Clothes – No Belt - Still	
4. Door Closed – Winter Clothes – No Belt - Movement	
5. Door Open – Winter Clothes – Belt - Still	
6. Door Open – Winter Clothes – Belt - Movement	
7. Door closed – Winter Clothes – Belt - Still	
8. Door Closed – Winter Clothes – Belt - Movement	
9. Door Open – Seat Empty	
10. Door Closed – Seat Empty	
Variations in Arrangements	**Code**
Person / Empty Seat	P / E
Closed Door / Open Door	C / O
Seat Belt / No Seat Belt	B / NB
Passenger Moving / Still	M / S
Example Use Case	**File Name**
Person - Door Closed – Winter Clothes – No Belt - Movement	P1CNBM



Example Use Case	File Name
Person - Door Open – Winter Clothes – Belt – No Movement	P1OBS



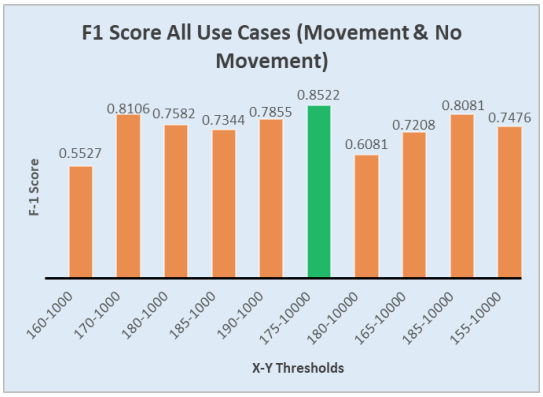
iv. Similar to the procedure we followed for creating the confusion matrix for Dataset 1, confusion matrix for Dataset 2 was created for all the use cases considered for the various thresholds and it was found that in comparison to confusion matrices of other thresholds, the one that was generated for the threshold [X, Y] = [175, 10000] showed the best value of F-1 score of 0.8522 as shown below:

		Predicted value: Positive	Predicted value: Negative		
Actual Value	Positive	2376	824	True positive rate (TPR) = 74.25%	False negative rate (FNR) ≈25.75 %
	Negative	0	3200	False positive rate (FPR) =0%	True negative rate (TNR) ≈1%
		Positive predictive value (PPV) ≈1 %	False omission rate (FOR) ≈20.47 %		
	Accuracy (ACC) =0.871	False discovery rate (FDR) = 0%	Negative predictive value (NPV) =79.52 %		
	F1 Score is	0.852223816			

	True positive (TP)
Class 1 is classified as Empty - **NEGATIVE**	False negative (FN)
Class 2 is classfied as Human - **POSITIVE**	False positive (FP)
	True negative (TN)

Figure 13: Confusion Matrix for 2nd Dataset when [X, Y] = [175, 10000] from Red Pitaya Sensor Classifier

v. The graph for resulting F-1 scores of all other thresholds is shown below:



From the Dataset 2 classification files for the threshold [X, Y] = [175, 10000], it was also observed that though the F-1 score was higher but for the use cases when there was 'No Movement' by the human subject then the number of 'False Negative' predictions were higher as compared to the cases in which the human subject was making movements and also for some 'Movement' cases, the False Negative prediction was even 0.

3. *Dataset 3*

i. For the 3rd Dataset, the following points were considered from the observations from the previous two dataset results:

- The angle of incidence of ultrasonic waves from the Red Pitaya sensor was varied by changing the vertical seat position (Tilting) and use cases were created around it.
- To analyze sensor's accuracy, now instead of an 'Empty Seat' use case, a 'Dummy Object' was placed wearing the same winter clothing which was also worn by the human test subject as shown below.
- The 'Door Open' use case was dropped since 'Door Closed' use case is more valid in real world environment.
- For all the new use cases, the 'Door Closed' and 'Seat Belt' factor was fixed and other arrangements were varied.

Figure 14: Dummy Object Wearing Winter Clothing with Belt and Seat Tilted (Front View)

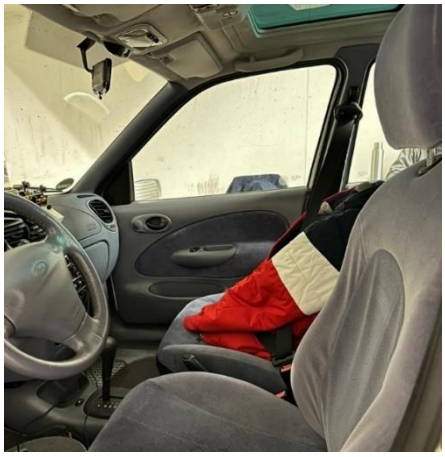


Figure 15: Dummy Object Wearing Winter Clothing with Belt and Seat Tilted (Side View from Driver's Seat)

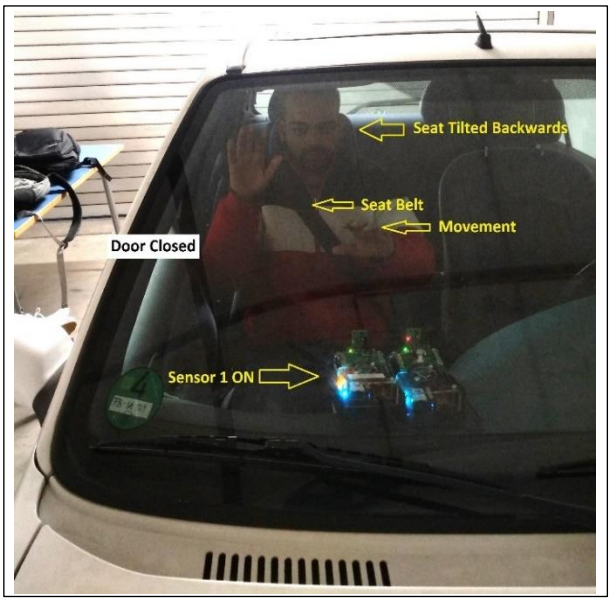
- The 'Door Open' use case was dropped since 'Door Closed' use case is more valid in real world environment.
- For all the new use cases, the 'Door Closed' and 'Seat Belt' factor was fixed, and other arrangements were varied.

ii. The below thresholds were used for all the use cases considered to get better classification results:

-t	X	Y
1.	60	1000
2.	80	1000
3.	100	1000
4.	120	1000
5.	140	1000
6.	160	1000

iii. Below table shows the six use-cases which were considered for every threshold mentioned above. It also describes an example use case:

Use Cases	
1. Door Closed - Winter Clothes – Belt - Still- Tilted	
2. Door Closed - Winter Clothes – Belt - Movement- Tilted	
3. Door Closed - Winter Clothes – Belt - Still - Not Tilted	
4. Door Closed - Winter Clothes – Belt - Movement - Not Tilted	
5. Door Closed - Winter Clothes - Dummy-Belt-Tilted	
6. Door Closed - Winter Clothes - Dummy-Belt-Not Tilted	
Variations in Arrangements	**Code**
Person / Dummy Object	P / E
Closed Door / Open Door	C / O
Seat Belt / No Seat Belt	B / NB
Seat Tilted / Not Tilted	T / NT
Passenger Moving / Still	M / S
Example Use Case	**File Name**
Person - Door Closed - Winter Clothes – Belt - Movement- Tilted	PC1BMT



iv. For all of the mentioned use cases confusion matrices were created under 3 categories for each of the thresholds to analyze the classier:

a. Confusion Matrix using All Use Cases (Movement, No Movement and Dressed Dummy).
b. Confusion Matrix using Only Movement and Dressed Dummy Use Cases.
c. Confusion Matrix using Only No Movement and Dressed Dummy Use Cases.

v. When All use cases are considered together **(Movement, No Movement and Dressed Dummy)**, it was found that the threshold **[X, Y] = [100, 1000]** gives the best value of **F1 score of 0.8148** among the rest of the considered thresholds. Figure (a) below shows the confusion matrix for the threshold [X, Y] = [100, 1000] and figure (b) shows the resulting F-1 scores of all thresholds for the '**Movement, No Movement and Dressed Dummy**' use cases:

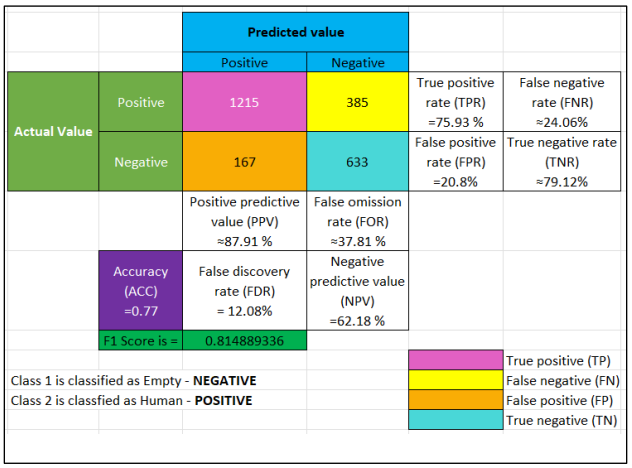
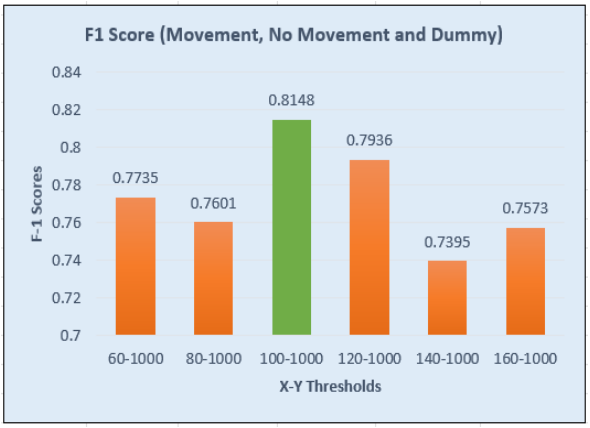


Figure 16: Confusion Matrix for threshold [X, Y] = [100, 1000] from Red Pitaya Sensor (Movement, No Movement and Dresses Dummy)



vi. For the **Movement Only** use cases, it was found that the threshold **[X, Y] = [120, 1000]** gives the best value of **F1 score of 0.9397** among the rest of the considered thresholds due to the fact for the 'Movement Only' use cases the number of incorrect predictions were significantly smaller as compared to 'No Movement' use cases. Figure (a) below shows the confusion matrix for the threshold [X, Y] = [120, 1000] and figure (b) shows the resulting F-1 scores of all thresholds for the '**Movement Only**' use cases:

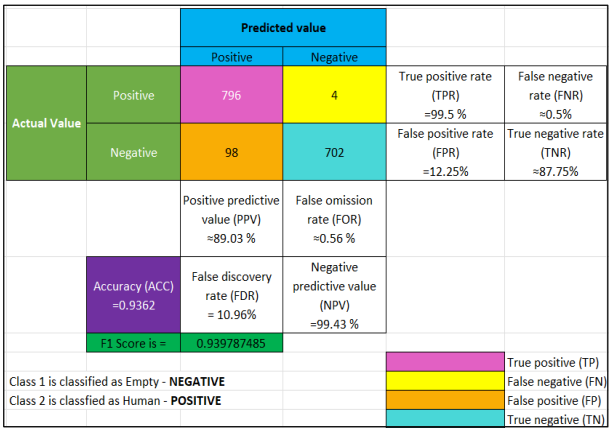
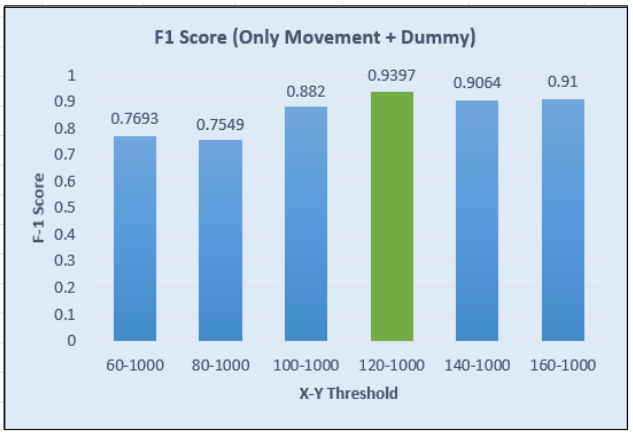


Figure 17: Confusion Matrix for threshold [X, Y] = [120, 1000] from Red Pitaya Sensor (Movement Only)



vii. For the **No Movement Only** Use Cases, it was found **F1 Score of 0.637** at the threshold **[X, Y] = [100, 1000]** but it is much lesser as compared with 'Movement Only' use case, which shows that the 'No Movement' use cases are not accurately classified by the classifier since the number of the number of incorrect predictions were significantly high and also the No Movement use cases are not practical in real world for a Human being sitting without any movement.

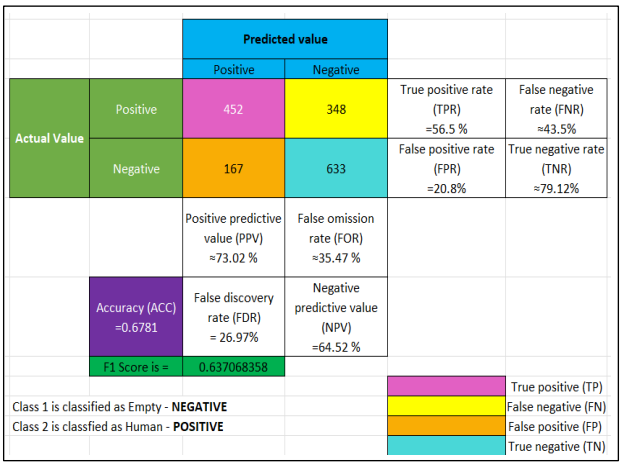
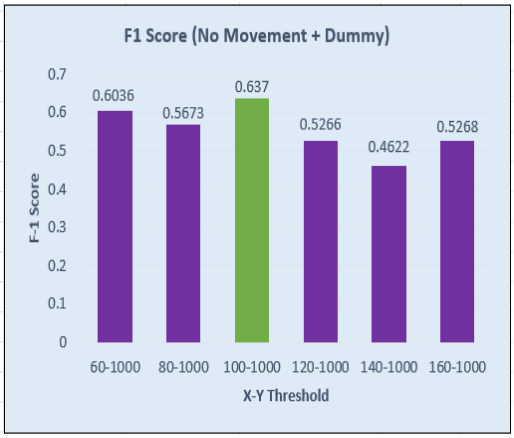


Figure 18: Confusion Matrix for threshold [X, Y] = [100, 1000] from Red Pitaya Sensor (No Movement)



C. *Development of a confusion matrix (without model)*

To generate a confusion matrix, the data set is collected in the second stage of the process. By doing so, the data captured may be understood more clearly, and it can be checked to see if the thresholds used to make predictions were accurate or not. In this case, Python (ver 2023.4.1) was used, and the actions below were taken to create confusion with a matrix:

1. Importing the packages

```
from io import StringIO
import os
import pandas as pd
import numpy as np
from sklearn import metrics
import seaborn as sns
import matplotlib.pyplot as plt

import csv
import openpyxl
```

2. Array declaration and initialization

```
Arr1_actual = []
arr1_predicted = []
```

3. Reading the 'actualValues.txt' and converting it into an array using the 'numpy' package

```
fa = open("actualValues.txt")
arr1_actual = np. loadtxt(fa, delimiter=",")
#print(arr1_actual)

fp = open("predictedValues.txt")
arr1_predicted = np. loadtxt(fp,
delimiter=",")
#print(arr1_predicted)
```

4. Building a Confusion matrix

```
##Confusion Matrix
cf_matrix =
metrics.confusion_matrix(arr1_actual,
arr1_predicted)
print(cf_matrix)
```

5. Rates and accuracy of the Confusion matrix

```
## Rates and accuracy of confusion matrix
p = np.count_nonzero(arr1_actual == 2)
n = np.count_nonzero(arr1_actual == 1)
pp = np.count_nonzero(arr1_predicted == 2)
pn = np.count_nonzero(arr1_predicted == 1)

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

accuracy = ((TP + TN)/(p + n)) #accuracy
f1score = ((2*TP)/(2*TP + FP + FN)) #F-1
Score

PPV = (TP/pp) #Positive predictive value
(PPV)
FDR = (FP/pp) #False discovery rate (FDR)
FOR = (FN/pn) #False omission rate (FOR)
NPV = (TN/pn) #Negative predictive value
(NPV)

TPR = (TP/p) #True positive rate (TPR)
FPR = (FP/n) #False positive rate (FPR)
FNR = (FN/p) #False negative rate (FNR)
TNR = (TN/n) #True negative rate (TNR)

print("\n Actual Positive(p): ",p)
print("\n Actual Negative(n): ",n)
print("\n Predicted Positive(pp): ",pp)
print("\n Predicted Negative(pn): ",pn)

print("\n\n True Negative(TN): ",TN)
print("\n False Positive(FP): ",FP)
print("\n False Negative(FN): ",FN)
print("\n True Positive(TP): ",TP)

print("\n\n accuracy: ",accuracy)
print("F1-Score: ",f1score)

print("\n\n      Positive      predictive
value(PPV): ",PPV)
print("\n False discovery rate(FDR): ",FDR)
print("\n False omission rate(FOR): ",FOR)
print("\n Negative predictive value(NPV):
",NPV)

print("\n\n True positive rate(TPR): ",TPR)
print("\n False positive rate(FPR): ",FPR)
print("\n False negative rate(FNR): ",FNR)
print("\n True negative rate(TNR): ",TNR)
```

6. Heat Map representation of the confusion matrix

```
ax = sns.heatmap(cf_matrix, annot=True,
cmap='Greens')

ax.set_title('Human or EmptySeat Detection
Confusion Matrix\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');
```

7. Ticket labelling – List must be in alphabetical order.

```
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
```

8. Displaying the visualization of the Confusion matrix

```
plt.show()
```

D. *Random Forest Classifier:*

To identify our data and make predictions in line with it, we are using the Random Forest Classifier. The model is built, the data is trained then confusion matrix, accuracy and F1 score are generated using the Python programming language (version 2023.4.1). The software program is broken down into the following steps:

1. Importing the Packages

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, f1_score
from sklearn.model_selection import train_test_split
from sklearn.tree import plot_tree
import pandas as pd
import matplotlib.pyplot as plt
from io import StringIO
import os
```

Figure 19: Importing necessary packages from Python Libraries

2. Among all the sets of readings received during a specific measurement slot, the 'Empty seat/ No Human' and 'Human' data set was read and segregated into two different files labeled 'empty.txt' and 'human.txt' respectively. 'Empty seat/ No Person' data set being read into 'empty.txt'

```
# Read in empty data from files
with open("empty.txt", "r") as f:
    lines = f.readlines()

empty = []
for line in lines:
    values = line.split()
    numValues = [int(x) for x in values]
    empty.append(numValues)
```

Figure 20: File read logic for 'Empty' set of data

3. The same goes for reading the data set for "Human" into "human.txt."

```
# Read in human data from files
with open("human.txt", "r") as f:
    lines = f.readlines()

human = []
for line in lines:
    values = line.split()
    numValues = [int(x) for x in values]
    human.append(numValues)
```

Figure 21: File read logic for 'Human' present set of data

4. The read data is now used to create a list of lists for the classifier's training and testing purposes. They are designated either "1" for "Empty" or "2" for "Human" by the labels.

```
# Combine data and labels
X = []
y = []
for line in empty:
    X.append(line)
    y.append(1)
for line in human:
    X.append(line)
    y.append(2)
```

Figure 22: List of lists creation for combining Data and Labels

5. The 'X' and 'y' arrays are now used to train the model. Please take note that the parameter 'test size=0.3' makes it clear that roughly 70% of the data is utilized to train the model and the remaining 30% to generate predictions.
In the code random state is set to 42. This is a common value used for random state in many machines learning libraries, including scikit-learn. random state is used to initialize the random number generator used by the Random Forest algorithm. By setting random state to a specific value, the algorithm will generate the same results every time the code is run. This can be useful for reproducibility and comparing different models.

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Figure 23: Splitting the data into 70% to be used for training the model

6. The parameters that are experimented with and set to achieve improved accuracy are then supplied to the Random Forest Classifier.

```
# Setting the hyperparameters of the Random Forest Classifier
n_estimators = 100
criterion = "gini"
max_depth = 4
min_samples_split = 2
min_samples_leaf = 1
random_state = 42
```

Figure 24: Setting the parameters of the Random Forest Classifier

7. The following command is used to train the model:

```
# Initializing and Train Random Forest classifier
clf = RandomForestClassifier(n_estimators=n_estimators,
                             criterion=criterion,
                             max_depth=max_depth,
                             min_samples_split=min_samples_split,
                             min_samples_leaf=min_samples_leaf,
                             random_state=random_state,
                             )
clf.fit(X_train, y_train)
```

Figure 25: Training the model

8. The model makes a prediction by doing the following,

```
# Make predictions on test set
pred = clf.predict(X_test)
```

Figure 26: Prediction by the model

9. The model generates the confusion matrix as below:

```
cf_matrix = confusion_matrix(y_test, pred)
```

Figure 27: Confusion matrix generation

10. The model calculates the accuracy as below,

```
accuracy = clf.score(X_test, y_test)
```

Figure 28: Calculating the accuracy

11. The following command is used to calculate the F1 score

```
f1 = f1_score(y_test, pred, average='weighted')
```

Figure 29: Calculating the F1- Score

12. Command to print the confusion matrix, F1 score, and accuracy is,

```
print("Confusion matrix:")
print(cf_matrix)
print("Accuracy:", accuracy)
print("F1 score:", f1)
```

Figure 30: Printing the confusion matrix

13. Command to visualize one of the decision trees in the random forest classifier.

```
# Visualize one of the decision trees in the random forest
plt.figure(figsize=(17,5))
plot_tree(clf.estimators_[0], class_names=['Empty', 'Human'], filled=True)
plt.show()
```

Figure 31: Visualization of decision tree

E. Random Forest Classifier Parameters Used:

The Random Forest Classifier is passed a few parameters to improve the predictions. The parameters used in this project are discussed below:

- **n_estimator: int, default=100**

The decision trees in the forest are indicated by the number n_estimators. The accuracy of the model may typically be increased by adding more trees, but doing so comes at a cost of higher computing complexity and longer training times. A popular range for n_estimators is between 50 and 500. Yet, the ideal choice for n_estimators may change based on the particular dataset and issue. The chosen value for this project is 100.

- **Criterion: *{'gini', 'entropy', 'log_loss'}, default= 'gini'***

It determines the function that is used to measure the quality of a split at each node in the decision trees. Supported criteria are 'gini' for the Gini impurity and 'log_loss' and 'entropy' both for the Shannon information gain. The value used for this parameter is 'gini'.

- **max_depth**: ***int, default=None***

The maximum number of splits that a decision tree can have. The model underfits the data if there are too few splits, and overfits the data if there are too many divides. We often stick to a maximum depth of 3, 5, or 7. This parameter has a value of '4' assigned to it.

- **min_samples_split: *int or float, default=2***

This is the minimum number of samples required to split an internal node in a decision tree. Setting a higher value can simplify the model and reduce the risk of overfitting but may also lead to underfitting. This parameter has a value of '2' assigned to it.

- **min_samples_leaf: *int or float, default=1***

In a decision tree, this is the bare minimum of samples needed at a leaf node. A higher value can make the model simpler and less prone to overfitting, but it can also result in underfitting. In regression in particular, this might result in a smoothing of the model. It is given the value of '1' for this project.

- **random_state: *int, RandomState instance or None, default=None***

It is used to set up the Random Forest algorithm's random number generator. The algorithm will produce the same results each time the code is executed if random state is set to a particular value. Comparing several models and ensuring reproducibility may benefit from this. Any integer can be used as random state's value, however it should always be fixed at one in order to have predictable results. The value '42' has been given to this parameter..

IV. Results from Random Forest Classifier

After using the Random Forest Classifier and generating the Confusion Matrix, Accuracy and F1 score for each of the Datasets in previous sections for every threshold value for 'All use cases', 'Movement Only' and 'No Movement Only' use cases, the followings results were obtained:

- The **Random Forest Classifier** model could predict that when the thresholds of **[X, Y] = [120, 1000]** were used, an accuracy of **0.995744** was obtained considering 'Only Movement' use case which is higher as compared to the accuracy of **0.9362** which was obtained for the same threshold for the same use case in case of **Red Pitaya classifier** as shown in the below table:

Table 3: Table showing the accuracy and F-1 Score

Classifiers	Accuracy	F1- Score
Red Pitaya Classifier Model	0.9362	0.9397
Random Forest Classifier Model	0.9957	0.9957

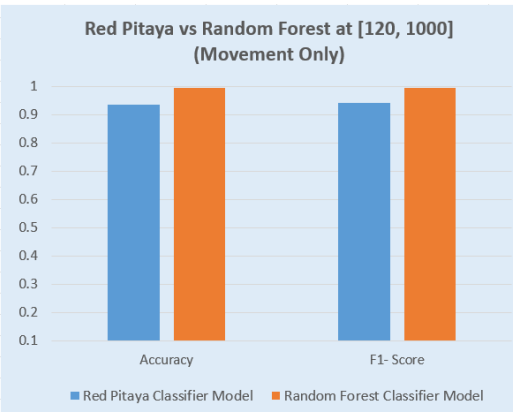


*Figure 32: Graph showing the comparison of Red Pitaya and Random Forest Classifiers at **[X, Y] = [120, 1000] (Movement Only)***

```
66   # Calculate and print confusion matrix, accuracy
67   cf_matrix = confusion_matrix(y_test, pred)
68
69   accuracy = clf.score(X_test, y_test)
70
71   f1 = f1_score(y_test, pred, average='weighted')
72
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

Confusion matrix:
[[229   1]
 [  1 239]]
Accuracy: 0.9957446808510638
F1 score: 0.9957446808510638
```

Figure 33: Confusion matrix, Accuracy and F1-Score for threshold X = 120, Y =1000 (generated by Random Forest Classifier for 'Movement' only cases)

- Similarly, for the threshold of **[X, Y] = [100, 1000]** the **Random Forest Classifier** model yielded the best accuracy of **0.99786** considering All use cases (Movement and No Movement) while the accuracy of Red Pitaya classifier model was **0.77** for the same scenario which is much lower than the accuracy obtained by Random Forest classifier model as shown in the below table:

*Table 4: Table showing the accuracy and F-1 Score for **[X, Y] = [100, 1000]***

Classifiers	Accuracy	F1- Score
Red Pitaya Classifier Model	0.77	0.8148
Random Forest Classifier Model	0.99786	0.99786

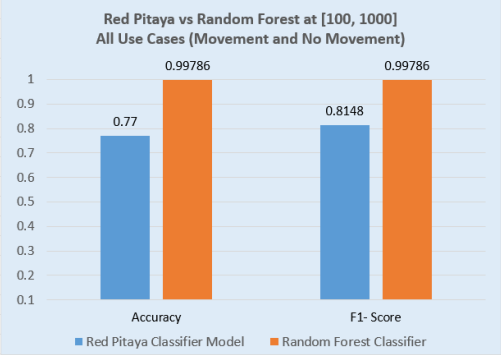


Figure 34: Graph showing the comparison of Red Pitaya and Random Forest Classifiers at [X, Y] = [100, 1000] for All Use Cases (Movement and No Movement)

```
66   # Calculate and print confusion matrix, accuracy
67   cf_matrix = confusion_matrix(y_test, pred)
68
69   accuracy = clf.score(X_test, y_test)
70
71   f1 = f1_score(y_test, pred, average='weighted')
72
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

Confusion matrix:
[[478   0]
 [  2 459]]
Accuracy: 0.9978700745473909
F1 score: 0.9978699827152596
```

Figure 35: Confusion matrix, Accuracy and F1-Score for threshold X = 100, Y =1000 (generated by Random Forest Classifier for All Use Cases (Movement and No Movement)

With the help of above analysis, we were able to draw important conclusions which are explained in the next section.

V. Analysis and Conclusion

The Random Forest Classifier outperforms the classifier found in the red pitaya sensor in terms of accuracy, and this is because it has more parameter values that may be adjusted to get greater F1 scores.

Also, we observed that the findings were significantly influenced by the layering of the winter clothing. The sensor was able to correctly classify the objects, and it had good accuracy when detecting "Empty" seats. Our studies led us to the additional conclusion that if there has been "No Movement" for more than a minute, the sensor should identify the space as vacant rather than a person. In this way, the fusion of sensor data with an additional warning signal for a moving car provided by an autonomous agent may be able to save a life. As a result of their interconnected nature, machine learning (ML) and autonomous intelligent systems (AIS) can assist make substantial progress in terms of applications and utilization.

VI. Acknowledgement

We are thankful of the constant direction we received from Prof. Dr. Andreas Pech and Prof. Peter Nauth as we carried out the project. Also, we are appreciative of the ongoing comments on the measurements that were made, which serve as a spur to pursue advancements and new understandings. In order to complete our project and reach our objective, we also want to thank the creators of the references we used and the web content.

VII. References

[1] S. Aggarwal, "Munich Personal RePEc Archive," 2023. [Online]. Available: https://mpra.ub.uni-muenchen.de/116579/.

[2] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," SN Computer Science, vol. 2, no. 3, p. 160, 2021.

[3] S. Das, A. Dey, A. Pal and N. Roy, "Applications of Artificial Intelligence in Machine Learning: Review and Prospect," International Journal of Computer Applications, vol. 115, pp. 31-41, 04 2015.

[4] V. Suganya, D. R. S, R. Priyadharshini.R, M. Ramana and S. R., "ENACTMENT OF ARTIFICIAL INTELLIGENCE IN MACHINE LEARNING WITH PRESUMPTION AND PANORAMA," WAFFEN-UND KOSTUMKUNDE JOURNAL, vol. 11, no. 4, pp. 321-328, April 2020.

[5] M. A. d. G. Maartje and F. M. Bertram, "How People Explain Action(and Autonomous Intelligent Systems Should Too)".

[6] S. Sah, "Machine Learning: A Review of Learning Types," 07 2020.

[7] K. Kaivan , "Deep Learning (Part 3) - Convolutional neural networks (CNN) (Galaxy Training Materials)," 24 05 2021.

[8] Y. Xiong, "Building text hierarchical structure by using confusion matrix," in 2012 5th International Conference on BioMedical Engineering and Informatics, 2012.

[9] Z. Karimi, "Confusion Matrix," 10 2021.

[10] W. Thomas, "F-Score," [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/f-score.

[11] M. A. Sovierzoski, . F. Mendes de Azevedo and F. I. M. Argoud, "Performance Evaluation of an ANN FF Classifier of Raw EEG

Data using ROC Analysis," in 2008 International Conference on BioMedical Engineering and Informatics, 2008, pp. 332-336.
[12] S. Saxena, "Deep dive into Confusion Matrix," TowardsAI , 2023. [Online]. Available: https://towardsai.net/p/l/deep-dive-into-confusion-matrix.
[13] Red Pitaya, [Online]. Available: https://redpitaya.com/.
[14] [Online]. Available: https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor.
[15] J. Ali, R. Khan, N. Ahmad and I. Maqsood, "Random Forests and Decision Trees," International Journal of Computer Science Issues(IJCSI), vol. 9, 09 2012.
[16] P. T R, "A Comparative Study on Decision Tree and Random Forest Using R Tool," IJARCCE, pp. 196-199, 01 2015.
[17] J. Sharma and R. Saxena, "Phishing Website Detection Using Machine Learning," IJCRT, vol. 10, no. 11, pp. 607-609, 2022.
[18] S. Kumar Gupta, "https://medium.com/@sarthakkumargupta/semi-supervised-and-supervised-machine-learning-750e2dade6d," [Online].
[19] "https://rollingrobots.com/arduino-ultrasonic-sensor," [Online].