Parameter Setting and Reliability Test of a Sensor System for Person Detection in a Car Wearing Winter Wear

Course M.Eng Information Technology

Modules Autonomous Intelligent Systems and Machine Learning

By Dr. Peter Nauth and Dr. Andreas Pech

Shiva Kumar Biru
Mat No: 1436929
shiva.biru@stud.fra-uas.de

Faiz Mohammad khan
Mat No: 1427310
faiz.khan@stud.fra-uas.de

Abstract : The critical need for accurate detection of individuals within vehicles, especially in winter conditions where thick clothing poses challenges. We investigate parameter settings and reliability testing of a sensor system tailored for this purpose. Our study aims to enhance safety and user experience in automotive systems by evaluating the system's performance in detecting individuals wearing winter clothing. Due to the complexity of these tasks, advanced machine learning algorithms have been developed to address these challenges. To train the machine learning model, RFC and SVM algorithms for supervised learning were opted. Accuracy and F1 score were examined by constructing confusion matrices for various scenarios, including both empty car seat and individuals wearing winter attire inside the vehicle.

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

I. INTRODUCTION

Our Study focuses on the accurate detection of individuals within a vehicle environment is of paramount importance for ensuring safety particularly in winter climates, where people often wear thick and different types of garments, the dependability and efficiency of sensor systems for detecting individuals become significantly more vital. It endeavors to thoroughly investigate the configuration of parameters and the testing of reliability concerning a sensor system designed specifically for identifying individuals within vehicles during winter conditions. The primary objective of this is to assess the performance and reliability of a sensor system designed to detect individuals wearing winter clothes within a car front seat with different jackets with different persons. To record the data, a sensor and the notion of Fourier transforms were utilized.

After adjusting the output readings and gathering the necessary datasets during the project, utilized these measurements to construct numerous confusion matrices for our study, employing supervised machine learning algorithms like Random Forest Classifier (RFC) and Support Vector Machines (SVM). Subsequently, we leveraged the outcomes of these matrices to further refine a machine learning model, aiming to enhance the accuracy and reliability of the sensor's output.

II. METHODOLOGY

The theoretical foundation of the experiment is outlined in the preceding section, encompassing explanations of the Ultrasonic Red Pitaya sensor, FFT and ADC data analysis methodologies, background information on Machine Learning algorithms, and the concept of Confusion matrices.

A. Ultrasonic sensor and Red Pitaya.

The Red Pitaya STEM Lab board integrates a system-on-a-chip (SoC) developed by Xilinx. In this configuration, Ultrasonic Sensor SRF02[Fig.2] was used, featuring a mean frequency of 40 kHz and a power output of 150 mW as per the manufacturer's specifications. Unlike dual transducer rangers, the SRF02 employs a single transducer for both transmission and reception, resulting in a minimum range of approximately 15 cm.

It operates with a 5V grounded power supply. The Red Pitaya embedded system [Fig.3], equipped with a sampling frequency of 1.95 MS/s and a resolution of 14 bits, controls the SRF02 and performs analog-to-digital conversion of the received signal. Moreover, the Red Pitaya device facilitates wireless data transfer to a laptop for further processing.

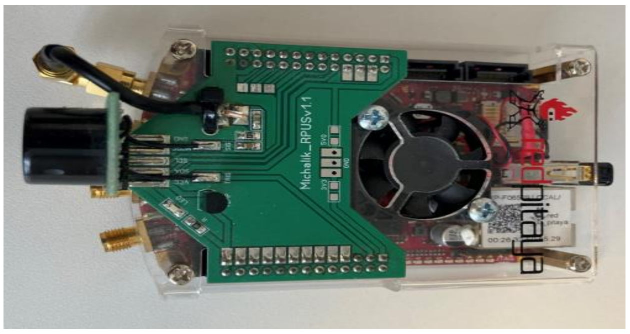


Fig.1. Ultrasonic sensor and red pitaya device

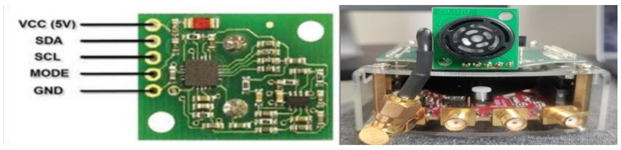


Fig.2. Sonar Sensor SRF02

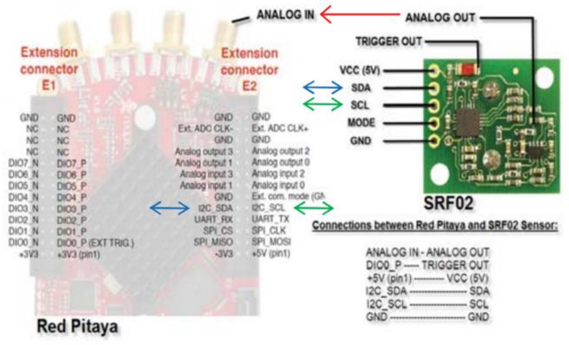


Fig.3.Interface between Embedded System and Sensor

It operates on the GNU/Linux operating system, enabling management and recording of measurements on a computer. Alongside 16 standard input and output ports, the primary Red Pitaya unit features two analog RF inputs and outputs. Additionally, it includes a micro-SD card slot, an RJ45 socket for Ethernet, a USB port, and a micro-USB connector for the console. The Red Pitaya can both receive and transmit radio frequency transmissions within the frequency range of 50MHz.

B. *Analog to Digital Converter*

Analog to Digital Converter is referred to as ADC. The Analog-to-Digital Converter (ADC) serves as a fundamental bridge between the continuous analog world and the discrete digital domain. With its capability to transform continuous analog signals into digital representations, the ADC enables the processing and analysis of signals through computational means.

In our study, the ADC data obtained from the Ultrasonic Red Pitaya sensor provides crucial insights into the characteristics of the signals captured. Through meticulous analysis of this digitized data, we aim to extract pertinent information regarding the environment under observation, particularly focusing on distance measurements. By harnessing the power of ADC data analysis techniques, such as signal filtering and sampling, the main objective was to enhance the accuracy and reliability of our sensor system.

C. *Fast Fourier Transform*

The Fast Fourier Transform (FFT) is an efficient algorithm used to compute the discrete Fourier transform (DFT) of a sequence, converting a signal from the time domain to the frequency domain. Employing a divide-and-conquer approach, the FFT method recursively splits a DFT of size N into two DFTs of size N/2 and combines the results in O(N) time. This results in significantly faster computation compared to the simple O(N^2) technique.

The FFT algorithm finds widespread application in various fields, including signal processing, data compression, image processing, and others, where analysis or manipulation of signals in the frequency domain is crucial. It has become a standard tool in numerous scientific and engineering disciplines.

D. *Confusion Matrix:*

A confusion matrix is a table used to evaluate the performance of a classification method by comparing the expected labels of a dataset with the actual labels. It serves as a standard tool in statistics and machine learning for assessing the accuracy of a classification model.

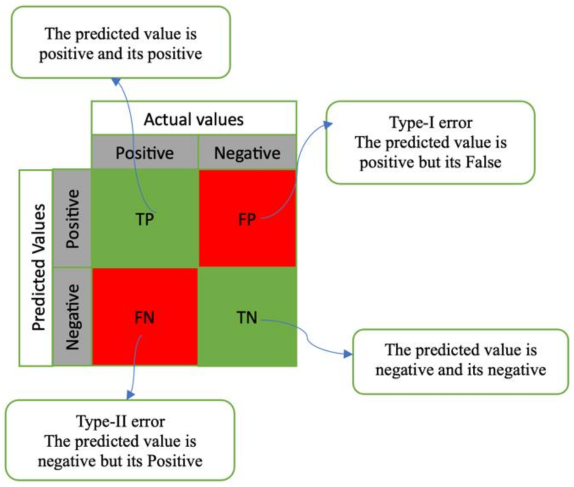

The predicted value is positive and its positive
Actual values
Positive
Negative
Predicted Values
Positive
Negative
TP
FP
FN
TN
Type-I error The predicted value is positive but its False
The predicted value is negative and its negative
Type-II error The predicted value is negative but its Positive

Fig.4. confusion matrix

The confusion matrix depicted in Figure 4 delineates various parameters derived from four fundamental metrics:

- **True Positive (TP)**: Occurs when the model accurately predicts a positive outcome, aligning with the actual positive value.

- **True Negative (TN):** Indicates instances where the model correctly predicts a negative outcome, corresponding to the actual negative value.

- **False Positive (FP):** Characterized by instances where the model erroneously predicts a positive outcome despite the actual value being negative, constituting the first type of prediction error. The FP is also known as Type-I error.
- **False Negative (FN):** Denotes cases where the model incorrectly predicts a negative outcome despite the actual value being positive, representing the second type of prediction error. These metrics are instrumental in computing additional parameters essential for evaluating the performance of the classification model. The FN is also known as Type-II error.

From the confusion matrix, various crucial metrics can be computed to assess the classification algorithm's performance, including:

Accuracy:

Accuracy represents the proportion of correctly predicted samples to the total dataset size, indicating the classifier's overall correctness. The accuracy formula is:

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

Recall (Sensitivity):

Recall measures the ratio of correctly predicted positive samples to the total number of positive samples in the dataset, demonstrating the model's effectiveness in capturing actual positives. The recall formula is:

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

Precision:

Precision quantifies the ratio of correctly predicted positive samples to the total number of positive samples predicted by the model, highlighting the model's ability to identify true positives. The precision formula is:

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

Specificity:

Specificity evaluates the ratio of correctly predicted negative samples to the total number of negative samples in the dataset, indicating the model's proficiency in identifying true negatives. The specificity formula is:

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

F1 Score:

The F1 score, representing the harmonic mean of precision and recall, offers a balanced metric when both precision and recall are essential. The F1 score formula is:

$$\text{F1 Score} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

The confusion matrix is instrumental in evaluating a classification model's effectiveness. It tabulates the occurrences of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) resulting from the model's predictions on a dataset. These figures allow for the computation of several performance rates, such as:

True Positive Rate (TPR) or Sensitivity:

TPR measures the proportion of actual positive cases that were correctly identified by the model as positive. It indicates how well the model identifies true positives among all actual positive cases. A high TPR value indicates that the model is good at correctly identifying positive cases. It is calculated as TP / (TP + FN).

False Positive Rate (FPR):

FPR measures the proportion of actual negative cases that were incorrectly classified as positive by the model. It indicates the model's tendency to label negative cases as positive falsely. A low FPR value suggests that the model is good at avoiding false alarms or incorrectly classifying negative cases.
It is calculated as FP / (TN + FP).

True Negative Rate (TNR) or Specificity:

TNR measures the proportion of actual negative cases that were correctly identified by the model as negative. It indicates how well the model identifies true negatives among all actual negative cases. A high TNR value suggests that the model is good at correctly identifying negative cases. It is calculated as TN / (TN + FP).

False Negative Rate (FNR):

FNR measures the proportion of actual positive cases that were incorrectly classified as negative by the model. It indicates the model's tendency to miss positive cases or incorrectly classify them as negative. A low FNR value suggests that the model is good at avoiding missing positive cases.
It is calculated as FN / (TP + FN).

These rates, encompassing TPR, FPR, TNR, and FNR, are crucial metrics for evaluating classification models. They provide a comprehensive understanding of how well a model distinguishes between positive and negative instances in a dataset. By analysing these rates, researchers and practitioners can assess a model's effectiveness and compare it with others. Moreover, understanding these rates allows for adjustments to improve performance and guiding optimization efforts. Overall, these metrics are essential for enhancing classification model performance and facilitating better decision-making across various applications.

E. Model Training:

Machine learning is a branch of artificial intelligence focused on developing algorithms and models that enable computers to learn from data and make predictions autonomously. Rather than relying on explicit programming, machine learning emphasizes learning from examples and experiences. These algorithms and models are trained on datasets containing input data and corresponding output values, aiming to predict output values for new input data. The training process involves adjusting model parameters iteratively to minimize the disparity between predicted and true outputs.

Machine learning encompasses various algorithms, including supervised, unsupervised, and reinforcement learning. In supervised learning, algorithms are trained on labelled datasets pairing input data with correct output values.

Unsupervised learning involves training on unlabelled data to uncover underlying patterns and structures. For our experiment, supervised learning was used, employing Support Vector Machine (SVM) and Random Forest classifiers (RFC) to create models and analyze the data.

F. Random Forest classifiers (RFC)

Random forests stand as a widely utilized supervised machine learning technique, applicable to scenarios with labelled target variables. Capable of addressing both regression tasks involving numeric target variables and classification challenges with categorical target variables, random forests offer versatility across various problem domains. Employing an ensemble approach, they amalgamate predictions from multiple individual models. Notably, each constituent model within the random forest ensemble operates as a decision tree, contributing to the collective predictive power of the algorithm.

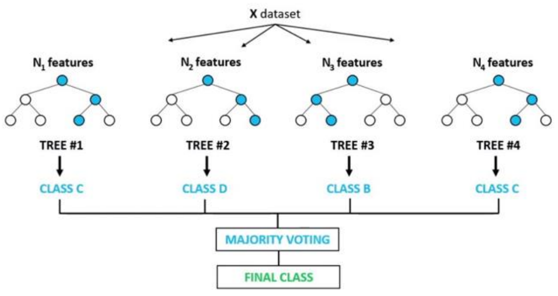


Fig.5. Random Forest classifiers Algorithm

The following steps are described to create a RFC model to detect a person wearing winterwear inside the car.

Step 1: Select random samples from a given data or training set.
Step 2: This algorithm will construct a decision tree for every training data.
Step 3: Voting will take place by averaging the decision tree.
Step 4: Finally, select the most voted prediction result as the final prediction result.

G. Support Vector Machine (SVM)

Support Vector Machine (SVM) stands out as a robust supervised learning algorithm utilized for classification and regression tasks. Its core principle involves identifying the optimal hyperplane within the feature space, effectively separating distinct classes while maximizing the margin between them. SVM exhibits efficacy in high-dimensional spaces and finds applications across diverse domains, such as image recognition, text classification, and bioinformatics.

SVM encompasses several variants:

1. **Linear SVM**: This variant employs a linear hyperplane to segregate classes, ideally suited for datasets exhibiting linear separability.

2. **Non-linear SVM (Kernel SVM):** Kernel SVM utilizes kernel functions to project input data into a higher-dimensional space, facilitating class separation through hyperplanes. Common kernel functions include Polynomial Kernel, Gaussian Radial Basis Function (RBF) Kernel, and Sigmoid Kernel.

3. **Multiclass SVM:** Despite being inherently binary, SVM can be extended to handle multi-class classification through techniques like One-vs-One and One-vs-All methods.

4. **Weighted SVM:** This variant assign varying weights to different classes, addressing issues related to class imbalance within the dataset

III. IMPLEMENTATION

A. Measurement Setup and Collection:

1) Measurement Environment:

For real-time measurements, the Ford Fiesta model, as depicted in Figure 6 was used. The car is parked in the basement garage at Frankfurt University of Applied Sciences, where all data measurements take place.

Fig.6. Ford Fiesta v16 Model used for measurement

The Red Pitaya Measurement board, along with the ultrasonic sensor, is installed on the dashboard in front of the side seat next to the driver's seat as shown in the figure 7

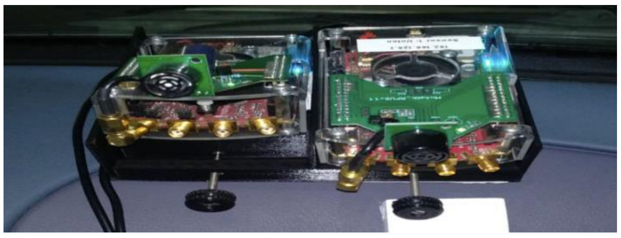


Fig.7. Red Pitaya Placement in the car

The setup of the setting for this experiment with the top view is illustrated in the figures 8 and 9 respectively. The sensor is placed on the dashboard which is100cm from the sensor to the passenger seat shown in figure 8 and 110cm from the sensor to the passenger seat shown in figure 9. The left FIUS sensor was used for the project. The side seat next to the driver's seat was the main area of measurement.

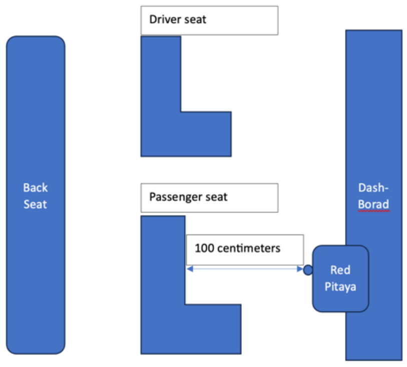


Fig.8. The Top view distance between sensor and the seat

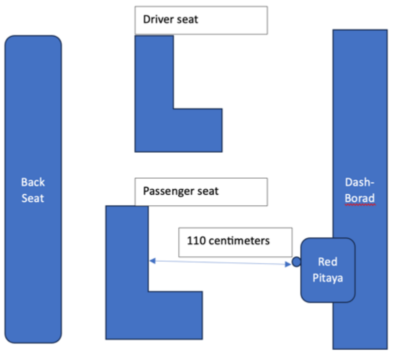
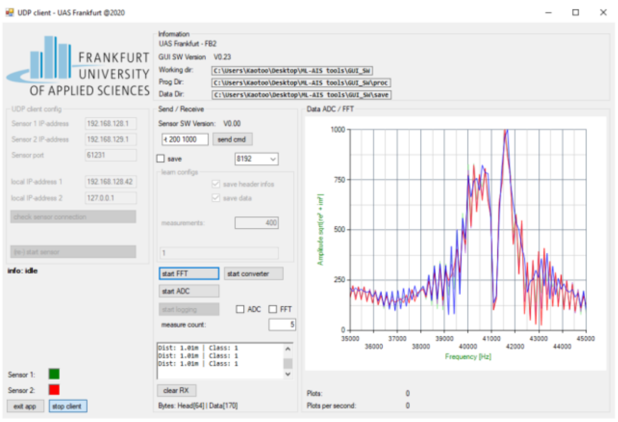


Fig.9. The top view distance between sensor and the seat

2) Measurement Software:

The data gathered from the Red Pitaya measurement board is processed utilizing measurement software created at Frankfurt University of Applied Sciences. Figure 10 below depicts the Graphical User Interface (GUI) of this measurement software. The software has been set up to analyze either the analog data or raw data from the ultrasonic sensor.



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

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

Sensor SW Version: V0.00
-t 200 1000
☐ save 8192
✓ save header infos
✓ save data
measurements: 400
1
☐ ADC ☐ FFT
measure count: 5

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

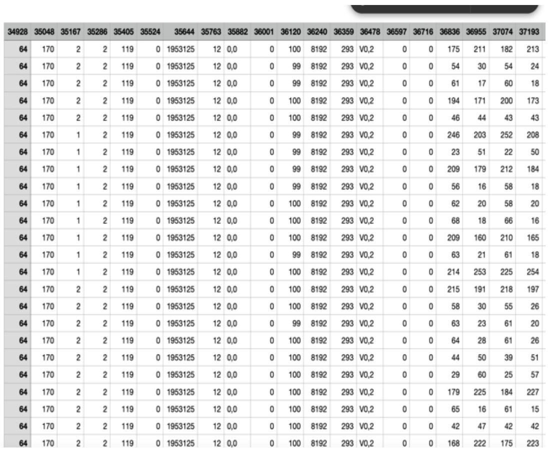
Bytes: Head[64] | Data[170]
Sensor 1: 🟩
Sensor 2: 🟥
Plots: 0
Plots per second: 0

Fig.10. FFT data from the measurement software

Next the raw data was subjected to the FFT and ADC and exported as FFT and ADC data. The software is typically used to gather data and study how it behaves in various contexts. It can also plot FFT and ADC and time-series graphs.

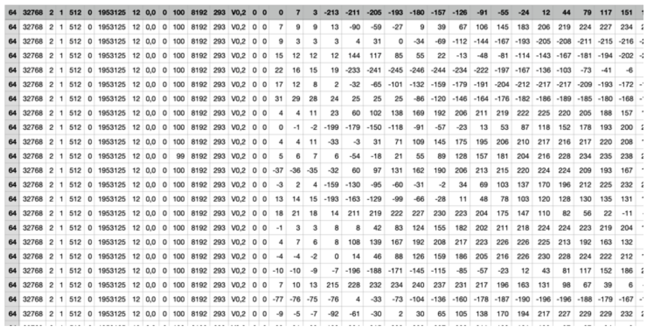
3) Data Format:

For this measurement, text-based FFT and ADC data that was exported from the software was used. Each FFT data has 85 columns overall, with rows representing amplitude values in the range of 0 to 1000 and columns representing frequency values in the range of 34.9 kHz to 44.9 kHz with intervals of 1.08 to 1.09 kHz shown in figure 10 and ADC data has 13400 columns overall with each row shown in figure 11(b).



34928	35048	35167	35286	35405	35524	35644	35763	35882	36001	36120	36240	36359	36478	36597	36716	36836	36955	37074	37193
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	175	211	182	213
64	170	2	2	119	0	1953125	12	0,0	0	99	8192	293	V0,2	0	0	54	30	54	24
64	170	2	2	119	0	1953125	12	0,0	0	99	8192	293	V0,2	0	0	61	17	60	18
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	194	171	200	173
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	46	44	43	43
64	170	1	2	119	0	1953125	12	0,0	0	99	8192	293	V0,2	0	0	246	203	252	208
64	170	1	2	119	0	1953125	12	0,0	0	99	8192	293	V0,2	0	0	23	51	22	50
64	170	1	2	119	0	1953125	12	0,0	0	99	8192	293	V0,2	0	0	209	179	212	184
64	170	1	2	119	0	1953125	12	0,0	0	99	8192	293	V0,2	0	0	56	16	58	18
64	170	1	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	62	20	58	20
64	170	1	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	68	18	66	16
64	170	1	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	209	160	210	165
64	170	1	2	119	0	1953125	12	0,0	0	99	8192	293	V0,2	0	0	63	21	61	18
64	170	1	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	214	253	225	254
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	215	191	218	197
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	58	30	55	26
64	170	2	2	119	0	1953125	12	0,0	0	99	8192	293	V0,2	0	0	63	23	61	20
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	64	28	61	26
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	44	50	39	51
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	29	60	25	57
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	179	225	184	227
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	65	16	61	15
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	42	47	42	42
64	170	2	2	119	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	168	222	175	223

Fig.11(a). Raw measurement data of FFT.



64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	0	7	3	-213	-211	-205	-193	-180	-157	-126	-91	-55	-24	12	44	79	117	151
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	7	9	9	13	-90	-59	-27	9	39	67	106	145	183	206	219	224	227	234
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	9	3	3	3	4	31	0	-34	-69	-112	-144	-167	-193	-205	-208	-211	-215	-216
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	15	12	12	12	144	117	85	55	22	-13	-48	-81	-114	-143	-167	-181	-194	-202
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	22	16	15	19	-233	-241	-245	-246	-244	-234	-222	-197	-167	-136	-103	-73	-41	-6
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	17	12	8	2	-32	-65	-101	-132	-159	-179	-191	-204	-212	-217	-217	-209	-193	-172
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	31	29	28	24	25	25	25	-86	-120	-146	-164	-176	-182	-186	-189	-185	-180	-168
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	4	4	11	23	60	102	138	169	192	206	211	219	222	225	220	205	188	157
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	0	-1	-2	-199	-179	-150	-118	-91	-57	-23	13	53	87	118	152	178	193	200
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	4	4	11	-33	-3	31	71	109	145	175	195	206	210	217	216	217	220	208
64	32768	2	1	512	0	1953125	12	0,0	0	99	8192	293	V0,2	0	0	5	6	7	6	-54	-18	21	55	89	128	157	181	204	216	228	234	235	238
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	-37	-36	-35	-32	60	97	131	162	190	206	213	215	220	224	224	209	193	167
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	-3	2	4	-159	-130	-95	-60	-31	-2	34	69	103	137	170	196	212	225	232
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	13	14	15	-193	-163	-129	-99	-66	-28	11	48	78	103	120	128	130	135	131
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	18	21	18	14	211	219	222	227	230	223	204	175	147	110	82	56	22	-11
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	-1	3	3	8	8	42	83	124	155	182	202	211	218	224	224	223	219	204
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	4	7	6	8	108	139	167	192	208	217	223	226	226	225	213	192	163	132
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	-4	-4	-2	0	14	46	88	126	159	186	205	216	226	230	228	224	222	212
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	-10	-10	-9	-7	-196	-188	-171	-145	-115	-85	-57	-23	12	43	81	117	152	186
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	7	10	13	215	228	232	234	240	237	231	217	196	163	131	98	67	39	6
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	-77	-76	-75	-76	4	-33	-73	-104	-136	-160	-178	-187	-190	-196	-196	-188	-179	-167
64	32768	2	1	512	0	1953125	12	0,0	0	100	8192	293	V0,2	0	0	-9	-5	-7	-92	-61	-30	2	30	65	105	138	170	194	217	227	229	229	232

Fig.11(b). Raw measurement data of ADC.

Data headers are also exported by the program in addition to FFT and 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 11(a) and 11(b) shows a sample data format which was received from the software and the table below, Table I, provides information on the data headers.

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	

B. Data Collection

With the setup finalized, the FFT and ADC measurement data is extracted from the Red Pitaya measurement board with an ultrasonic sensor, detecting individuals or objects. We establish specific conditions to observe differences across all scenarios. Our approach to implementing the model's response is iterative, based on ongoing analysis and feedback. We delve into detailed techniques and enhancements aimed at optimizing our model and refining the output throughout various phases. Data collection occurred in two phases:

- Empty, movement, and no movement scenarios using winter wear without threshold values with 100cm and 110cm the seat distanced from the sensor.
- Empty, movement, and no movement scenarios using winter wear with threshold values with 100cm and 110cm the seat distanced from the sensor.

In the baseline scenario for our study, we meticulously configured the car seat to be positioned 100 cm away from the Red Pitaya sensor. This setup ensured consistent and controlled conditions for our measurements. Throughout our experimentation, we meticulously documented various scenarios encountered during passenger occupancy within the vehicle.

Firstly, measurements were recorded for the empty seat car where no passenger is present on the car seat and then started recordings with the passenger in different scenarios. Measurements were recorded as passengers entered the car, capturing any variations in distance as they approached the seat. Subsequently, readings were taken while passengers remained seated in a standard position, providing a baseline for static measurements. We then observed the sensor's response to dynamic changes in distance as passengers engaged in movements such as adjusting seatbelts or reaching for items. Additionally, we evaluated the sensor's ability to detect subtle changes in distance as passengers shifted slightly forward while maintaining a constant position. Furthermore, sensor's performance was tested when passengers sat with crossed positioned, both in static and dynamic scenarios, including conversing with hand movements. Finally, measurements were taken as passengers exited the vehicle, monitoring changes in distance as they moved away from the sensor. These meticulously documented scenarios serve as crucial components of our study, enabling a comprehensive assessment of the sensor's accuracy and reliability in real-world conditions encountered during real-world conditions encountered during passenger occupancy in the car.

The below figures show the scenarios.

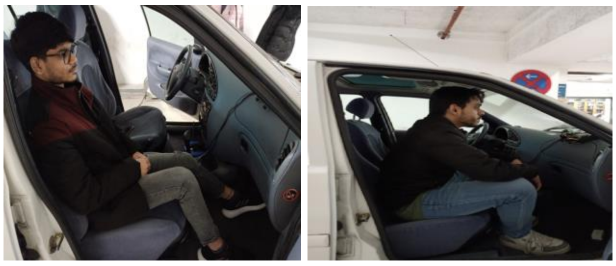


Fig.12. passenger seated normally and near to the sensor.

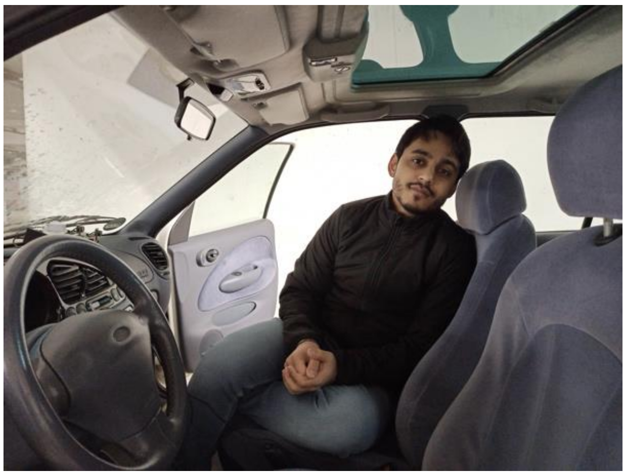


Fig.13. passenger cross seated.

Same as the baseline scenario, but the seat is leaned slightly, and all the above scenarios are covered.

The below images show the scenarios.

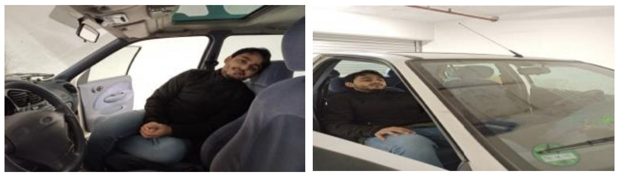


Fig.14. passenger normal and cross seated on a leaned seat.

The experiment is carried out with different jackets with different persons.

C. *Random Forest Classifier implementation*

Utilizing the Random Forest Classifier algorithm proves to be a strong approach for training models using data collected from the specified sensor setup. This algorithm employs ensemble learning, creating numerous decision trees during the training process. Each tree is trained on a random subset of the data, which helps prevent overfitting and enhances the model's ability to make accurate predictions on new data. By combining the predictions of these individual trees, the algorithm generates a final prediction that typically demonstrates improved accuracy and reliability compared to using a single decision tree.[4]

Additionally, Random Forest Classifier is adept at handling high-dimensional data, making it well-suited for the complex, multi-dimensional nature of sensor data. These attributes make Random Forest Classifier a compelling choice for training models designed to analyze data obtained from the Red-Pitaya measurement board equipped with an ultrasonic sensor.

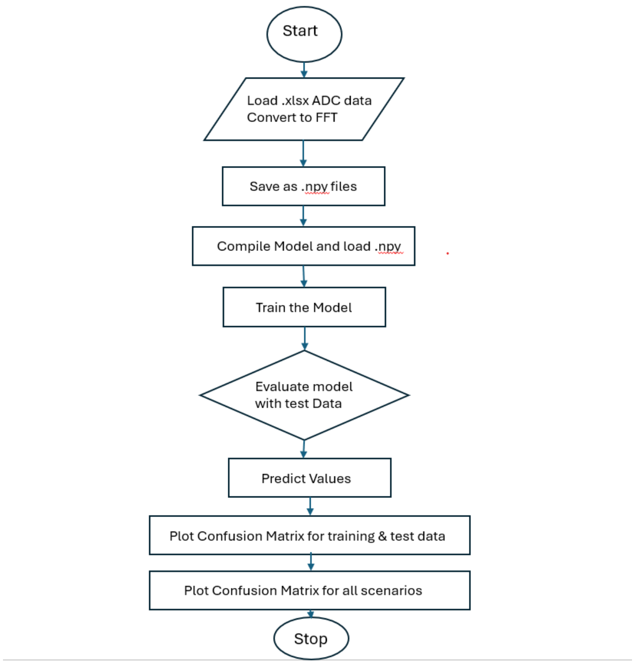

Start
Load .xlsx ADC data Convert to FFT
Save as .npy files
Compile Model and load .npy
Train the Model
Evaluate model with test Data
Predict Values
Plot Confusion Matrix for training & test data
Plot Confusion Matrix for all scenarios
Stop

Fig.15. Flow Chart of model

1) Data Pre-processing:

When the data is first extracted from the Red-Pitaya measuring board, which has an ultrasonic sensor, it is saved in the.txt format then converted to .xlsx. Then, using the power of the Python modules Pandas and NumPy, a conversion procedure is used to turn the data into .npy files, which makes the rest of the process easier. After this conversion, the dataset is divided into two parts: 80% of the data is used for training the model, and the remaining 20% is kept for testing and prediction operations.

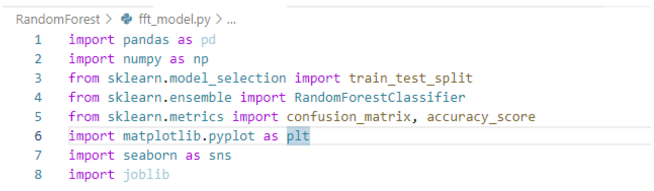
The 'Class Detected' header parameter that is taken from the.txt files is essential throughout the data labelling phase. Class 1 items are those that are inflexible, and Class 2 features or people, as shown in Table I under the 'Class Detected' Header. Making use of this data, the training set of .npy files is further divided into two subsets: the Human wearing winter apparel class and the Empty class. These subsets serve as the foundation for the model building and compilation efforts that come after.

2) Building and Training Random Forest Classifier:

The Random Forest Classifier is part of Python's sklearn library and can be imported from *sklearn.ensemble*. Popular Python machine learning toolkit scikit-learn, often known as sklearn, offers effective tools for data mining and analysis, including a number of different algorithms for dimensionality reduction, clustering, regression, and classification. Building upon NumPy, SciPy, and matplotlib, its straightforward and effective design makes it very popular.

Numpy is used to load the numpy arrays (.npy) which were saved from the .xlsx data.

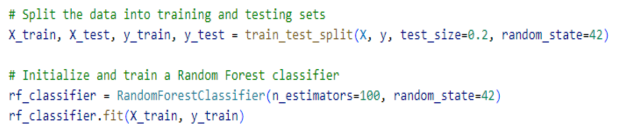
From *skelarn.metrics*, the functions *confusion_matrix* and *accuracy_score* are used to build the confusion matrix and calculate the accuracy of the training data



```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
```

Fig.16. Python Modules used for RFC model

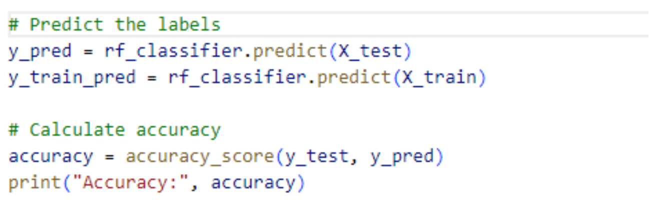
The data is loaded and split into training and testing data. The model is defined and configured to create an ensemble of 100 decision trees (***n_estimators=100***). The ***random_state=42*** parameter ensures that the randomness in the algorithm's behavior is reproducible, as it sets the seed for the random number generator. This means that running the algorithm with the same ***random_state*** value will produce the same results, aiding in result reproducibility and debugging.



```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train a Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
```

Fig.17. Training the Model

Once the Model is trained with our training data, labels were predicted for the Testing data and calculate the accuracy of the true positive values.



```
# Predict the labels
y_pred = rf_classifier.predict(X_test)
y_train_pred = rf_classifier.predict(X_train)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

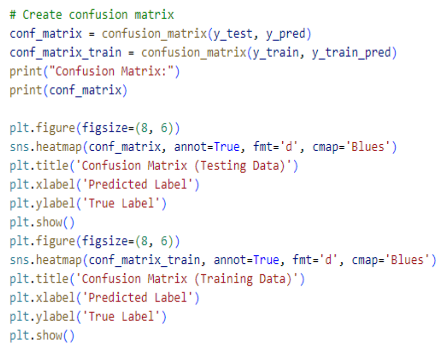
Fig.18: Predicting the labels and calculating Accuracy.

3) Plotting the Confusion Matrices

The python libraries matplotlib and seaborn are used to plotting the confusion matrices.

Matplotlib is a comprehensive library for creating static, interactive, and animated visualizations in Python. It offers a wide range of plotting functions and customization options, making it suitable for various data visualization tasks.

Seaborn is built on top of Matplotlib and provides a higher-level interface for creating attractive and informative statistical graphics. It simplifies the process of creating complex visualizations by offering built-in themes and functions for common statistical plots.

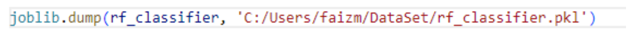


```
# Create confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
conf_matrix_train = confusion_matrix(y_train, y_train_pred)
print("Confusion Matrix:")
print(conf_matrix)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix (Testing Data)')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_train, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix (Training Data)')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

Fig.19. Confusion matrix generation

The confusion matrix for both the testing data and the training data is plotted, although we are more interested in the confusion matrix of the testing data to observe the model's capability to predict the right labels.

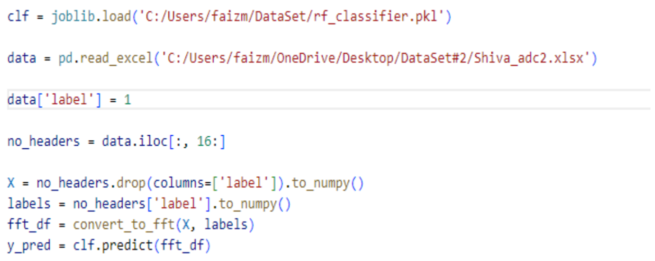
Finally we save the classifier as a .pkl file using the joblib library.



```
joblib.dump(rf_classifier, 'C:/Users/faizm/DataSet/rf_classifier.pkl')
```

4) Plotting the Confusion Matrix for Each Scenario

For scenario wise data which is present in the dataset, we load the saved classifier and use it to predict the values for data belonging to a particular scenario or test case. The scenario could be different persons or different jackets or different seat positions



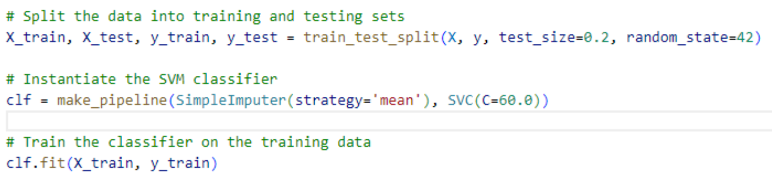
```
clf = joblib.load('C:/Users/faizm/DataSet/rf_classifier.pkl')

data = pd.read_excel('C:/Users/faizm/OneDrive/Desktop/DataSet#2/Shiva_adc2.xlsx')

data['label'] = 1

no_headers = data.iloc[:, 16:]

X = no_headers.drop(columns=['label']).to_numpy()
labels = no_headers['label'].to_numpy()
fft_df = convert_to_fft(X, labels)
y_pred = clf.predict(fft_df)
```

Fig.20.The Confusion Matrix for Each Scenario

D. SVM Implementation

We have initially used SVM Machine Learning model to train our Dataset 1, however the efficiency of this model was not the best and there was a problem of overfitting thus we moved to Random Forest Classifier instead.

SVM operates by locating the hyperplane in the feature space that best divides various classes. In order to increase the model's capacity for generalisation, its goal is to maximise the margin between the classes. By using several kernel functions, including linear, polynomial, and radial basis function (RBF) kernels, SVM can handle both linear and non-linear classification tasks. SVM is frequently utilised because of its efficiency, adaptability, and capacity to handle high-dimensional data in a variety of areas.



```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Instantiate the SVM classifier
clf = make_pipeline(SimpleImputer(strategy='mean'), SVC(C=60.0))

# Train the classifier on the training data
clf.fit(X_train, y_train)
```

Fig.21. Training the Model using SVM Classifier

Using scikit-learn's `*make_pipeline*` function, this code creates a pipeline of pre-processing steps that are followed by the SVM classifier, thereby instantiating an SVM (Support Vector Machine) classifier. In this pipeline:

- The mean value along each column is used by {SimpleImputer(strategy='mean')} to impute missing values in the dataset.
- {SVC(C=60.0)} sets the regularisation parameter {C} to 60.0 when the SVM classifier is first initialised. This parameter manages the trade-off between maintaining the model's tiny weights (coefficients) to prevent overfitting and obtaining a low training error.

The rest of the methodology remains the same as we used in the RandomForestClassifier Implementation.

IV. RESULTS AND DISCUSSION

In total, three datasets were created by reading the Sensor values (The link to all datasets can be found at the end of the Results and Discussion section). Each Dataset contains the different scenarios with different people and different Jackets. The Confusion matrix and associated metrics, which were produced for the test data using the RandomForestClassifier model and the trained SVM, are explained in this section. We take into consideration an event data for the test data, which is defined as an individual wearing winter clothing entering the car, sitting there for a while constantly, moving, leaning the seat and sitting cross, and then exiting the vehicle.

Each person wearing a jacket, regardless of whether they were sitting still or moving around a lot, was used as a testcase for the model. The testcases were then changed by altering the orientation and position of the seats with various features. The testcase situation as well as the quantity of ADC data used are described in the table below.

TABLE 2 . TEST DATA SCENARIOS

Test Case	Jacket	Seat Distance	Action
TestCase 1	Jacket1	100 cms	Const+Move
TestCase 2	Jacket2	110 cms	Constant
TestCase 3	Jacket3	110 cms	Constant
TestCase 4	Jacket2	110 cms	Moving
TestCase 5	Jacket3	110 cms	Moving

Each such test case was applied to three different people with the same jackets.

Label 0 represents no passenger.

Label 1 represents passenger wearing Winter Wear

A. Without Threshold (Dataset 1) – SVM Model

In the initial phase of our investigation, **Dataset#1**, which comprises a diverse range of scenarios involving individuals wearing various types of winter wear within a vehicle, served as a critical foundation for our exploratory analysis. This dataset, meticulously collected without applying any threshold adjustments, aimed to capture the raw essence and variability inherent in real-world conditions. The primary objective was to examine the effectiveness of the Support Vector Machine (SVM) model in discerning the presence of passengers, leveraging the model's capacity for handling high-dimensional data and its prowess in classification tasks.

Utilizing the SVM model, we trained Dataset#1 under the premise that its sophisticated algorithm would adeptly navigate through the intricacies of the dataset's features. Upon completion of the training phase, the SVM model demonstrated an impressive accuracy of 97.7%, suggesting a superficially robust capability in passenger detection scenarios. Such a high accuracy metric initially presented an optimistic outlook towards the potential application of SVM models in enhancing vehicle safety systems, particularly in the challenging conditions posed by winter attire.

However, a deeper analysis and scrutiny into the model's performance revealed significant concerns regarding overfitting. Overfitting occurs when a model learns the training data to such an extent that it negatively impacts its ability to generalize to new, unseen data. This phenomenon was evidenced in our project through the disparity between the model's training accuracy and its performance on validation sets.

To visually articulate these findings, we generated confusion matrices, which serve as a graphical representation of the model's performance across different scenarios within Dataset#1. The confusion matrices highlighted the distribution of true positives, true negatives, false positives, and false negatives, providing insight into the model's predictive capabilities and its tendency towards overfitting.

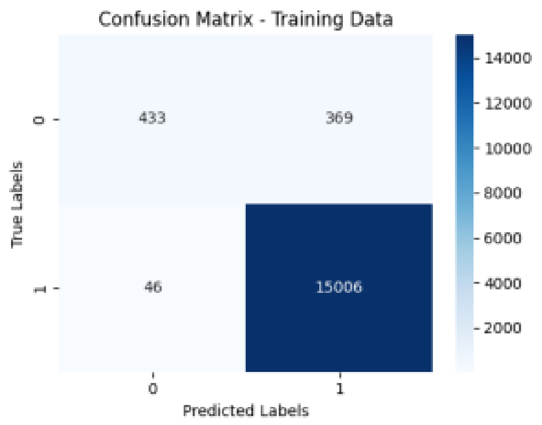


Figure 22: Training Dataset#1

This image should showcase the confusion matrix derived from the training data, illustrating the model's high accuracy and the minimal occurrence of false positives and false negatives within the training set.

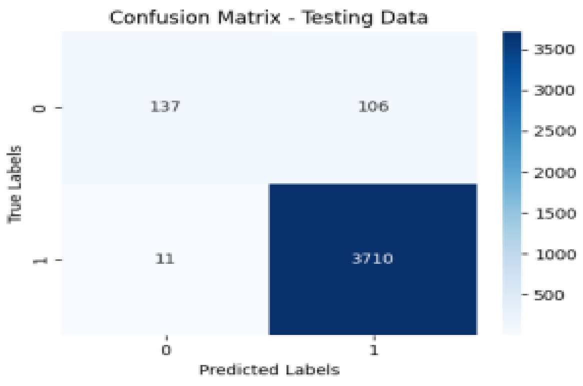


Figure 23: Testing Dataset#1

Conversely, this image should present the confusion matrix for the validation data, where the effects of overfitting become apparent through a less favourable distribution of predictive outcomes, notably an increase in false positives and false negatives when compared to the training data.

The stark contrast between the performance metrics of the training and validation datasets underscores the reliability issues associated with the SVM model trained on Dataset#1 without threshold adjustments. Despite the high accuracy figure, the practical applicability of such a model is considerably diminished due to its overfitting, underscoring the necessity for more nuanced approaches in model training and validation to ensure robust and reliable detection of passengers wearing winter clothing.

B. *With Threshold (Dataset 2) – RandomForestClassifier*

In our continued exploration to refine the detection of individuals within vehicles under winter conditions, we transitioned to **Dataset#2** for our subsequent phase of analysis. This dataset encompassed the same diverse scenarios as Dataset#1, but with a crucial modification in our approach: the implementation of threshold adjustments for sensor readings, specifically setting a threshold of 160 for feature 1 and 9000 for feature 10. This strategic change aimed to enhance the quality and discriminative power of the data, facilitating more accurate and reliable model training and predictions.

For this phase, we employed the RandomForestClassifier model, renowned for its robustness and ability to handle high-dimensional data through an ensemble of decision trees. The RandomForestClassifier was chosen with the anticipation that its inherent mechanisms for reducing overfitting—by averaging over multiple decision trees—would yield a model that not only performed well on training data but also generalized effectively to new, unseen data.

Upon training Dataset#2 with the RandomForestClassifier, we achieved an accuracy of 79.14%. While at first glance, this figure may seem modest compared to the previously obtained 97.7% accuracy with the SVM model, it's important to delve deeper into the context behind these numbers. The RandomForestClassifier's accuracy reflects a balanced trade-off between learning the training data and maintaining the model's ability to generalize. This balance is crucial in real-world applications where the model will encounter data variations not present in the training set.

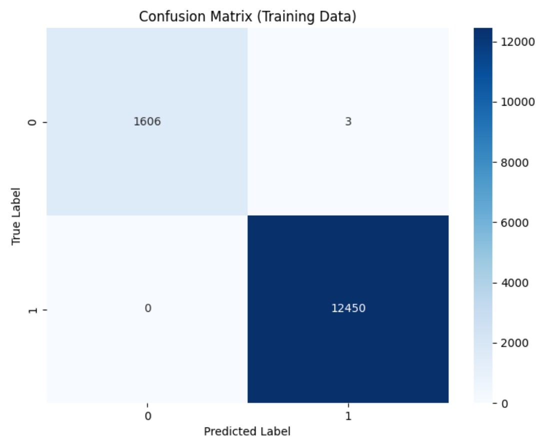


Figure 24: Training Dataset#2

This image should depict the confusion matrix resulting from the training data. It will highlight the balanced performance of the RandomForestClassifier, showcasing its adeptness at correctly classifying true positives and true negatives without succumbing to overfitting.

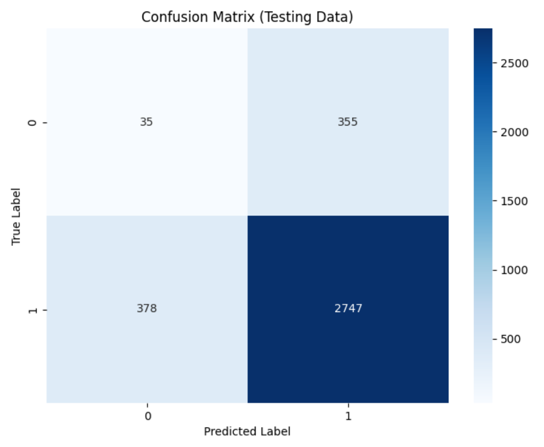


Figure 25: Testing Dataset#2

This second image should present the confusion matrix for the validation data. Unlike the previous SVM model, here we expect to see a more consistent performance between training and validation datasets, indicating the model's reliability and its successful generalization to new data.

The accuracy of **79.14%** achieved with the RandomForestClassifier, in conjunction with the strategic threshold adjustments in Dataset#2, marks a significant milestone in our project. This approach not only mitigated the risk of overfitting but also underscored the importance of preprocessing and model selection in developing reliable machine learning systems for passenger detection. The RandomForestClassifier's performance, supported by the threshold-based sensor data preprocessing, provides a promising avenue for enhancing the accuracy and reliability of vehicle safety systems, especially in the challenging scenarios presented by winter wear. This model's success illustrates the potential of machine learning algorithms, when appropriately harnessed, to contribute meaningfully to automotive safety innovations.

C. With Threshold Car Engine On (Dataset 3)

The RandomForestClassifier's performance after being trained on Dataset #3 is displayed in the confusion matrix below. As with Dataset #2, deliberate threshold modifications were made to improve the data quality, with a threshold of 160 for feature 1 and 9000 for feature 10 being maintained. This increase shows how well the RandomForestClassifier handles fluctuations in data, even in simulated scenarios instead of real-world ones.

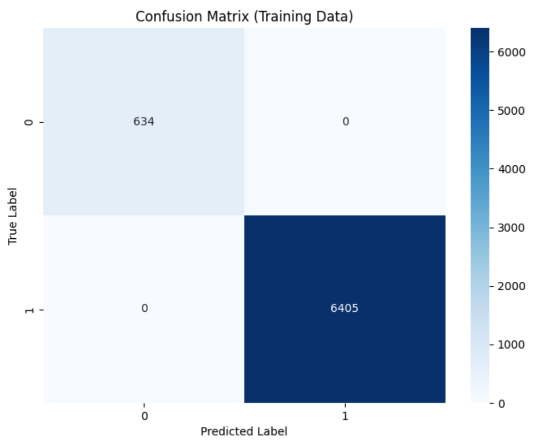


Figure 26: Training Dataset#3

The confusion matrix for Dataset #3's validation data is shown in Figure 27. The RandomForestClassifier exhibits consistent performance across the training and validation datasets, despite the hand simulation of a car engine running through shaking. With test data, the model outperformed Dataset #2 in terms of accuracy, achieving **83.97%.** This reinforces the model's dependability and good generalisation to new data.

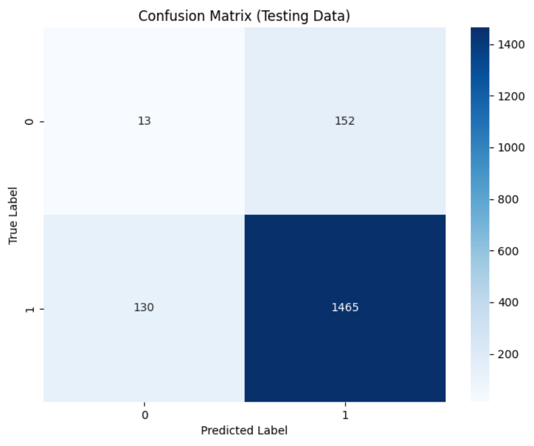


Figure 27: Test Dataset#3

The improved accuracy achieved with the RandomForestClassifier on Dataset#3, in comparison to Dataset#2, combined with the strategic threshold adjustments and manual simulation of car engine running, further solidifies the effectiveness of our approach in mitigating overfitting and improving model generalization. This emphasizes the importance of preprocessing techniques and model selection in developing robust machine learning systems for passenger detection, particularly in challenging conditions such as winter environments. The success of the RandomForestClassifier in conjunction with Dataset#3 offers promising prospects for enhancing the accuracy and reliability of vehicle safety systems, contributing significantly to automotive safety advancements.

D. Scenario Wise Results against dataset with threshold

It was found that passenger moving, or seated constant did not give much of a difference in accuracy, the confusion matrices plotted for the data weather the passenger was constantly moving or seated constantly was similar after setting the threshold of (-t 160 9000), The data shown below is from Dataset#2 and was predicted using the RandomForestClassifier.

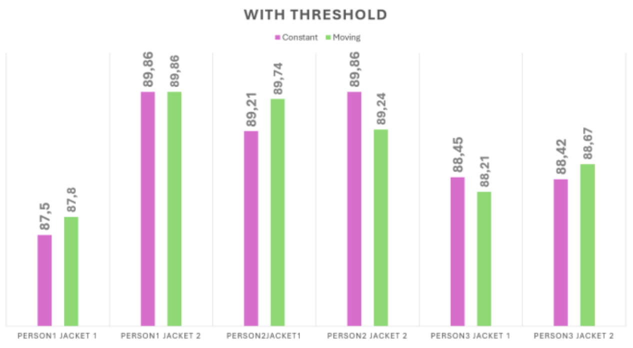


Figure 28: Scenario Wise Accuracy

Example confusion matrices can be seen in the figure below for passenger one wearing a specific Jacket.

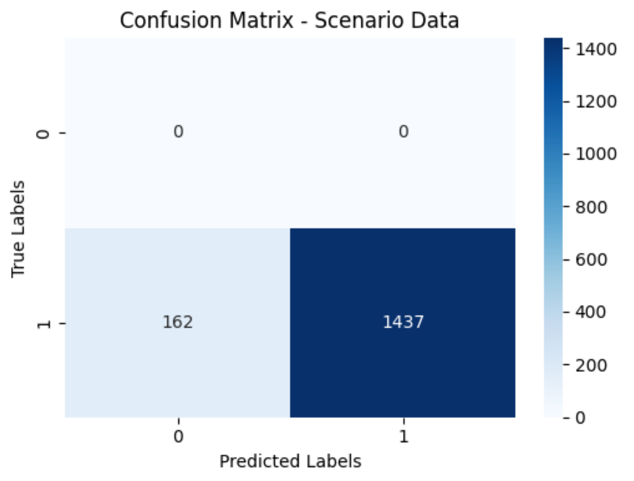


Figure 29: Person1 Jacket2 Constant

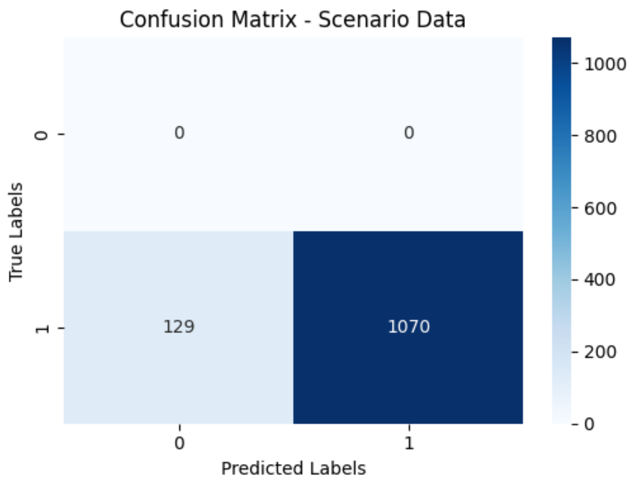


Figure 30: Person1 Jacket2 Moving

D . Scenario wise comparison results with the thresholds

Results were compared across four distinct scenarios: Constant Normal, Moving Normal, Constant Leaned, and Moving Leaned. Within each scenario, assessments were made for both distances, 100cm and 110cm, with and without the application of a threshold. This structured methodology enabled a thorough investigation of performance outcomes, considering variations in distance and the presence of thresholds within the specified scenarios.

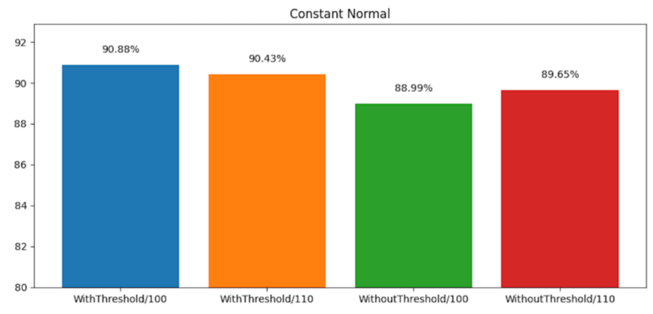


Figure 31: Person seated in normal seat with constant position

The bar chart titled 'Constant Normal' compares the performance outcomes under four different conditions. Performance is measured in percentages, with two conditions including a predefined threshold(Without Threshold) and the second threshold “ -t 160 9000”(With Threshold). Each condition is further examined with two different parameter values, denoted by '/100' and '/110.' The results indicate a marginal decrease in performance when the threshold is not applied, as seen in the green and red bars, compared to when it is applied, as represented by the blue and orange bars. These findings suggest that the application of a threshold has a positive impact on the performance of the system or process being analysed.

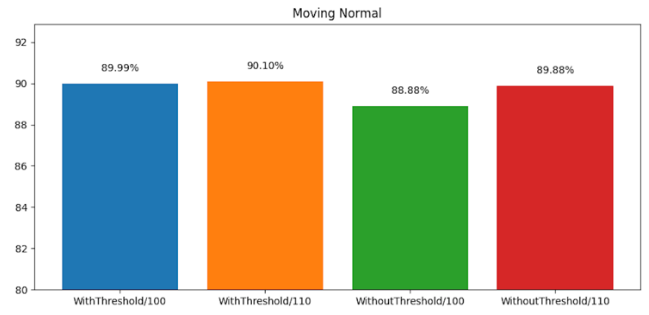


Figure 32: Person seated in normal seat with moving position.

The bar chart titled 'Moving Normal' compares the performance outcomes under four different conditions. Performance is measured in percentages, with two conditions including a predefined threshold (Without Threshold) and the second threshold “**-t 160 9000**” (With Threshold). Each condition is further examined with two different parameter values, denoted by '/100' and '/110.' The results indicate a marginal decrease in performance when the threshold is not applied, as seen in the green and red bars, compared to when it is applied, as represented by the blue and orange bars. These findings suggest that the application of a threshold has a positive impact on the performance of the system or process being analyzed.

Similarly for the moving leaned and the constant leaned we observe the same result as of the Normal constant and the normal moving with slight vary in the percentage accuracy as shown in the *figure 33* and *figure 34* respectively.

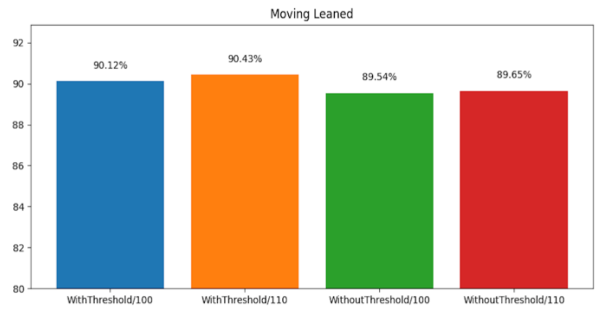


Figure 33: Person seated in normal seat with constant position.

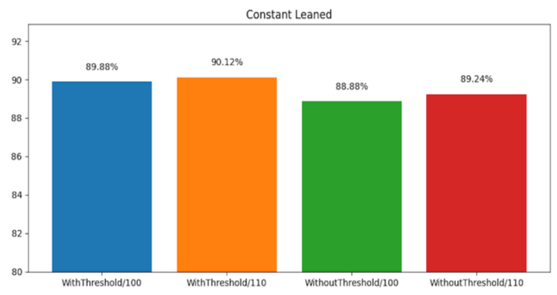


Figure 34: Person seated in normal seat with constant position.

LINK TO ALL DATASETS: All Datasets

V. DISCUSSION

The RandomForestClassifier outperformed the SVM model by offering a more balanced, generalizable, and interpretable solution to the complex problem of detecting passengers wearing winter clothing in vehicles. Its resilience to overfitting, adeptness at managing high-dimensional data, compatibility with our data pre-processing strategy, and the added benefit of model interpretability collectively made RandomForestClassifier the better choice for our project.

Comparison with Previous Milestones:

The transition from SVM to RandomForestClassifier, along with refined threshold adjustments, led to improved accuracy and model robustness. Early milestones indicated overfitting with SVM, which was mitigated by the ensemble approach of RandomForestClassifier.

Reason for false predictions:

The analysis aims to differentiate between scenarios, such as the presence of passengers with winter wear, by discerning unique patterns correlated with passenger occupancy. However, false predictions arise from factors such as environmental noise, material property variations affecting ultrasonic wave absorption and reflection, and the utilization of machine learning algorithms like Random Forest Classifier and Support Vector Machines (SVM). These predictions are evaluated through performance metrics derived from confusion matrices, including accuracy, precision, recall, and F1 score. False predictions stem from issues such as overfitting or underfitting of machine learning models, inadequate feature representation, and class imbalance in the dataset, where models fail to generalize to unseen data, capture underlying patterns effectively, or differentiate between classes accurately, respectively, thus impacting the reliability of passenger detection systems.

Findings:

The high accuracy and reliability of the RandomForestClassifier in detecting passengers wearing winter clothing have significant implications for automotive safety systems.

VI. CONCLUSION

This project successfully demonstrated the application of machine learning algorithms and the use of Autonomous Intelligence System specifically the FIUS Ultrasonic Sensor, in accurately detecting passengers wearing winter clothing in a vehicle. The findings underscore the potential of advanced data pre-processing and machine learning models in enhancing vehicle safety systems, with the proposed methodology showing promising results in varied test scenarios.

VII. ACKNOWLEDGMENT

We would like to express our profound gratitude to Professor Dr. Andreas Pech and Professor Dr. Peter Nauth for their important supervision and assistance during our work on this issue. They also helped us to finish the research and this report by pointing us in the proper direction. The "Autonomous Intelligence System" and "Machine Learning" lectures were very helpful in providing us with the inspiration and background information we needed to write this seminar report. We also acknowledge Prof. Julian Umansky for his important assistance in providing the necessary resources needed to finish the assignment successfully.

VIII. 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] J. Ali, Random Forests and Decision Trees, https://www.researchgate.net, 2012.

[5] H. Z. S. a. M. H. Hayder Hasan, A Comparison Between Support Vector Machine (SVM) and Convolutional Neural Network (CNN) Models For Hyperspectral Image Classification, 2019.

[6] K. Balani, S. Deshpande, R. Nair and V. Rane, "Human detection for autonomous vehicles," IEEE International Transportation Electrification Conference (ITEC), 2015. [Online]. Available: doi: 10.1109/ITEC-India.2015.7386891..

[7] A. H. Pech, P. M. Nauth and R. Michalik, "A new Approach for Pedestrian Detection in Vehicles by Ultrasonic Signal Analysis," IEEE EUROCON 2019 - 18th International Conference on Smart Technologies, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8861933.