Discrimination of reflected sound signals

Master of Engineering in Information Technology
Machine Learning
C. Chadha, M. Majid, S. Sharma
1324510, 1324374, 1324507
chetan.chadha@stud.fra-uas.de,
maria.majid@stud.fra-uas.de,
sanjana.sharma@stud.fra-uas.de

***Abstract*—Machine learning (ML) in the acoustics and signal processing domain has experienced rapid advancements and developments with persuasive outcomes over a course of years. The statistical techniques of Machine Learning offer detection of data patterns, which helps in the identification of the convoluted relationship between features and further discrimination, or classification based on these features. One of the ML techniques, called Binary classification is usually used to discriminate between two class labels. This project provides an ML-based solution for the discrimination of reflected sound signals, which are reflected from two different objects. Firstly, data pre-processing is performed on the reflected time signals to render the dataset. Secondly, Quadratic Time-Frequency representation (QTFR) of the reflected sound signal is generated and features extraction is performed on it. Afterward, four different Machine Learning classification models; namely, K-Nearest Neighbors, Random Forest, Logistic Regression, and Decision Trees are utilized for data training and prediction for the realization of binary classifier or discriminator. Finally, an assessment of classification results based on accuracy and various other measures are presented and discussed.**

Keywords—Machine Learning, Quadratic Time-Frequency Representation, Binary Classification, Discriminator, K-Nearest Neighbors, Random Forest, Logistic Regression, Decision Trees

I. INTRODUCTION

The recent inventions of smart and intelligent systems and devices have increased the utilization of Machine Learning and computational intelligence-based algorithms. A wide range of research on the popular topic of Machine Learning (ML) algorithms and techniques are available, which have brought considerable progress and ease to routine activities in this digital era. Machines are being trained to perform the task that humans do. The innovative algorithms and techniques of Machine learning (ML) have empowered advancements in automatic data processing and pattern recognition across many sciences and engineering fields. ML offers intelligence-based solutions to complex engineering challenges, in the same manner as the processing of the human brain. Additionally, ML deals with a variety of diverse big datasets such as image, video, audio, time-series signals, 1D signals, text, etc, which are vastly produced and stored by intelligent systems [1]. ML techniques in the domain of sound signals and acoustics have gained much attention for its persuasive solutions towards crucial tasks such as identification and validation of different sound signals. These ML algorithm detects data patterns by extracting useful attributes and features from the given dataset. Data labeling is performed afterward by utilizing these patterns. ML-based statistical methods enable the system or machine to learn and predict based on pattern recognition [2].

ML classification technique works through estimating the mapping pattern that logs the training dataset to the target class or label [3]. One of the ML classification techniques which acquire only two class labels is referred to as the Binary classification technique. The input dataset samples are classified into two states by computing specific classification measurements. There are 2 disjoint classes available for binary classification [4]. Some of the popular ML algorithms that can be used to realize binary classifier are Logistic Regression, k-Nearest Neighbors (k-NN), Decision Trees, Support Vector Machine (SVM), Naive Bayes, etc.

The inspiration behind this study is to deliver a classification-based solution to discriminate between the reflected sound signals coming from different objects. The considered use-case scenario is the acoustic signals that incident on the surface of some objects and reflects back. The time signal is formed after recording those reflected sound signals. The resultant time signal is the convolution of the incident sound waveform along with the reflecting object's surface properties. By analyzing the properties of the reflected time signal, the knowledge about the reflecting object can be achieved, on the basis of which the discrimination between reflected time signals is possible.

The ML-based solution for the aforementioned challenge is the main goal of this study. The aim is to realize a Binary Classifier or discriminator by applying different ML algorithms on the labeled dataset. The employed ML algorithms include Logistic Regression, k-Nearest Neighbors (k-NN), Decision Trees, and Random Forest (RF). This Binary classifier or discriminator is used to classify or distinguish the reflected sound signals, which belong to two different objects, named Object#1 and Object#2. Before implementing the binary classification algorithm, the Quadratic time-frequency representation is created for given sound signals, based on which the associated features are extracted to perform classification.

This project report presents an ML-powered solution for the discrimination of reflected sound signals by using the binary classification technique. The framework of this paper is as follows: Section II provides a brief description of the utilized techniques for the realization of the Binary classification or Discriminator model. Section III illustrates the workflow, step-by-step approach, and methodology to achieve the discriminator of the reflected signal. Section IV explains the python-based code implementation for the binary classification model and its GUI application. In Section V, the result analysis and assessment for the implemented binary classification model based on some evaluation metrics and predicted results by GUI application will be discussed. Section VI defines the guidelines to rebuild the proposed solution Finally, the project report is concluded in Section VII.

II. TECHNIQUES USED IN BINARY CLASSIFICATION MODEL IMPLEMENTATION

Before explaining the main implementation approach, a brief overview of the techniques employed in achieving the Binary Classification model is presented in this section.

A. Quadratic Time-Frequency Representation (QTFR)

The Quadratic Time-Frequency representation (QTFR) of a signal is quite effective for the evaluation of a time-varying signal [5]. The time-frequency representation is called a "spectrum" or a "spectrogram" since a quadratic time-frequency representation roughly represents the energy density of a signal in the time-frequency domain. For the evaluation of The QTFR spectrogram, some distinct features can be extracted from the spectrogram for further detection, estimation, and classification purpose. There are various methods available to analyze the QTFR of a signal, out of which the Short-time Fourier Transform (STFT) method is utilized broadly in many applications [6]. The available methods for the evaluation of the QTFR of a signal are as follows.

- Short-time Fourier Transform (STFT) Spectrogram (belongs to Cohen's class)
- Wigner-Ville distribution (WVD)
- Choi-Williams distribution (CWD)
- Cone-shaped distribution (CSD)
- Gabor spectrogram
- Adaptive spectrogram

1) STFT Spectrogram: The STFT works on the linear method to implement the time-frequency representation of a signal. The STFT is obtained by passing the signal through the bandpass filter bank that has a constant bandwidth [7]. The normalized, squared modulus of the STFT is the short-time Fourier transform (STFT) spectrogram. This spectrogram of the signal is the Bilinear or Quadratic time-frequency representation of the signal. Since the STFT spectrogram obeys "Parseval's energy-conservation theorem", therefore the energy in the STFT spectrogram is equal to the energy in the original time-varying signal [8]. The STFT spectrogram encompasses different properties and distinct features that can be extracted for the purpose of classification.

Following are some of the STFT spectrogram properties that are used in this project for the feature extraction phase.

a) Maximum Frequency of Spectrogram: From the STFT spectrogram of "one" time signal, the maximum value or peak of the signal's spectrum, denoted as Maximum Frequency, can be computed as a feature for further classification.

b) Spectrum Energy: The energy of the spectrum can be evaluated by summing up all elements of the STFT spectrogram of "one" time signal.

c) Instantaneous Power of Spectrogram: The instantaneous power of the spectrogram can be computed by dividing the maximum amplitude of the spectrogram by the given time instant.

d) Average Energy of Spectrogram: The average energy of the spectrogram can be calculated by dividing the spectrum energy with the length of the spectrum.

e) Mean Instantaneous Frequency: The mean frequency of a signal represents the "center of gravity" of a signal's spectrogram [9]. The mean frequency which relies on time variations is called the mean instantaneous frequency, which can be calculated by using (1).

$$\omega_t = \frac{\int_{-\infty}^{\infty} \omega\, SP(t,\omega) d\omega}{\int_{-\infty}^{\infty} SP(t,\omega) d\omega} \quad (1)$$

Where, $SP(t,\omega)$ is the spectrogram of a signal.

f) Group Delay: The Group delay represents the time delay of a time-varying signal in terms of frequency. The Group delay can also be used to calculate the propagation time of a signal across the system as a frequency function *[9]*. The group delay can be computed by using (2).

$$Group\ delay = \frac{\int_{-\infty}^{\infty} t\, SP(t,\omega) d\omega}{\int_{-\infty}^{\infty} SP(t,\omega) d\omega} \quad (2)$$

Where, $SP(t,\omega)$ is the spectrogram of a signal.

An example of STFT spectrogram-based quadratic time-frequency representation of a signal is depicted in Fig 1.

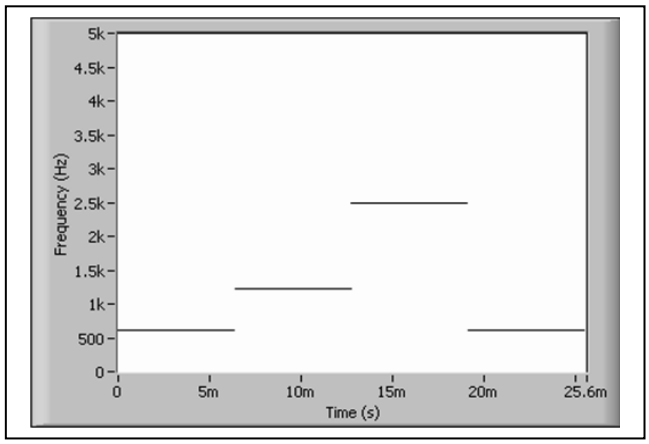

Fig. 1. STFT spectrogram example

2) Wigner-Ville Distribution (WVD): The WVD method utilizes a variation of autocorrelation function called instantaneous autocorrelation, to obtain the Power spectrum of a signal [8]. The WVD can be computed using (3).

$$W_s(t,\omega) = \frac{1}{2\pi} \int s\left(t + {}^{\tau}/_{2}\right) s^*\left(t - {}^{\tau}/_{2}\right) e^{-j\omega\tau} d\tau \quad (3)$$

Where τ is the time lag and $*$ is the complex conjugate of the signal s.

In comparison to the STFT spectrogram, the signal in WVD is a shifted version of itself. It is obtained by comparing the parameters of the signal with its own parameters at different times and frequency values [8]. An example of a WVD-based quadratic time-frequency representation of a signal is depicted in Fig 2 [10].

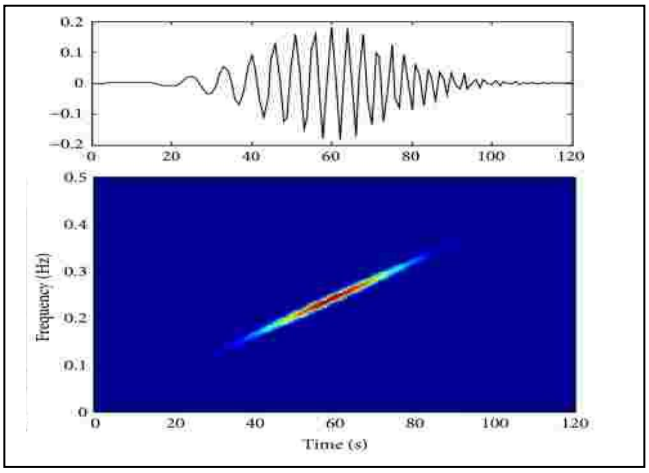

Fig. 2. WVD-based QTFR example

B. *The Binary Classification Technique*

Machine learning (ML) provides a set of techniques and algorithms that learn from examples. Classification is a task that necessitates the use of machine learning algorithms to learn how to apply a class or label for a specific problem. In machine learning, there are several different types of classification tasks to be encountered, as well as advanced modeling methods to be used for each. There are four key types of classification tasks that are listed as follows [11].

- Binary Classification
- Multi-Class Classification
- Multi-Label Classification
- Imbalanced Classification

In the Binary Classification technique, one class represents the "true" state and the other represents the "false" state. Class label 1 is assigned to the true state class, and the class label 0 is given to the false state class. The prediction technique used in the Binary classification model is based on Bernoulli's probability distribution. The Bernoulli distribution is a discrete probability distribution that describes a situation in which an occurrence has a binary outcome of 0 or 1. This means that the model estimates the likelihood of an example falling into class 1, or class 0 [11]. Fig. 3 depicts the scatter plot generated using the Binary classification model, in which two distinct clusters for two different classes can be observed.

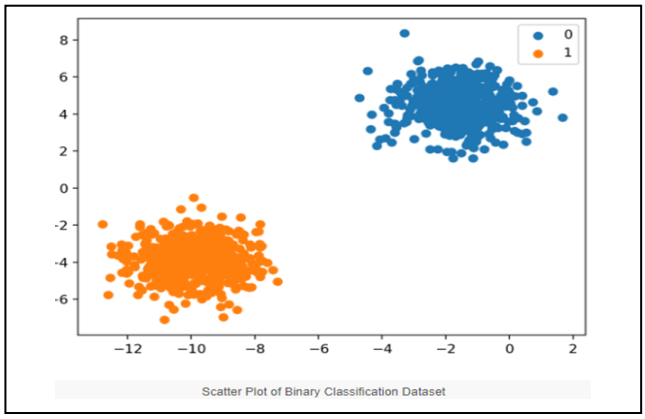

Fig. 3. Scatter plot example of the Binary classification model

Many ML classification algorithms can be used to realize the Binary classification model. The employed classification techniques used in this project include Logistic Regression, k-Nearest Neighbors (k-NN), Decision Tree, and Random Forest (RF).

1) Logistic Regression: Logistic Regression, is a relatively simple and efficient *ML* algorithm for deciding between two classes. It effectively creates a function that acts as a boundary between two groups. To perform binary classification, the Logistic Regression algorithm is broadly used. It employs the Logistic or Sigmoid function to forecast the likelihood that the problem's solution will be 1 or 0, yes or no, true, or false, etc. In a Logistic regression algorithm, input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) (y) to predict an output value, as demonstrated in (4). The output value being modeled is binary value 0 or 1 [12].

$$y = \frac{e^{(b_0+b_1*x)}}{1+e^{(b_0+b_1*x)}} \quad (4)$$

Where y is the expected output, b_0 is the bias or intercept term, and b_1 is the single input value coefficient (x).

The logistic regression algorithm's coefficients (Beta values b) must be calculated from the training dataset. Maximum-likelihood estimation is used for this. A pictorial representation of the Logistic regression classification is illustrated in Fig. 4.

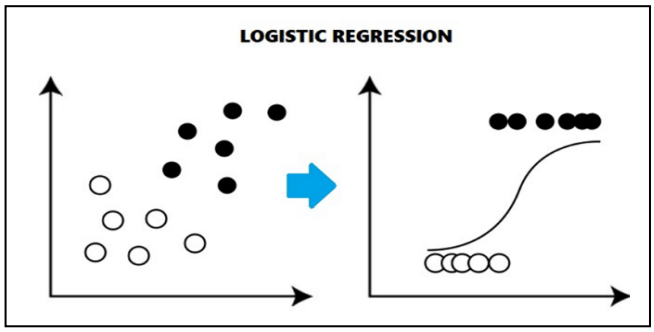

Fig. 4. Classification example for Logistic Regression model

2) k-Nearest Neighbors (k-NN): k-NN is the simplest, yet lazy supervised ML classification technique. It performs the computation when it is needed, which makes the k-NN classifier a lazy learner. In the classification phase of the test data point from the provided training dataset, firstly k-NN learns the training data set, performs its computation on it, and then predicts the class. The classification using the k-NN algorithm is done in the following steps [13], which are explained with Fig. 5 [14].

- Distance computation with each vector point.
- Determine nearest neighbors.
- Class comparison and voting for determined nearest neighbors.

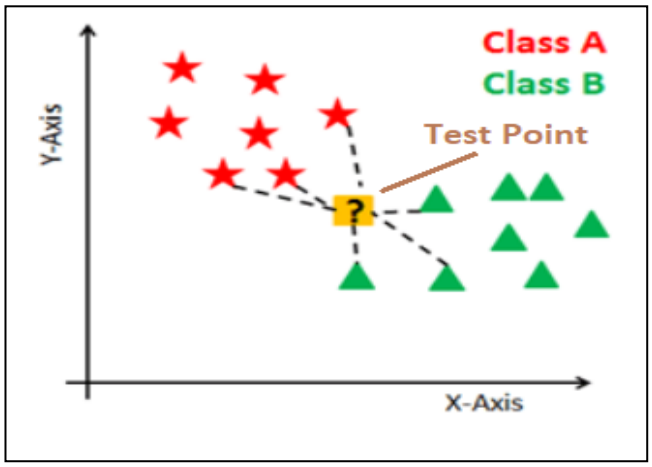

Fig. 5. Classification example for k-NN model

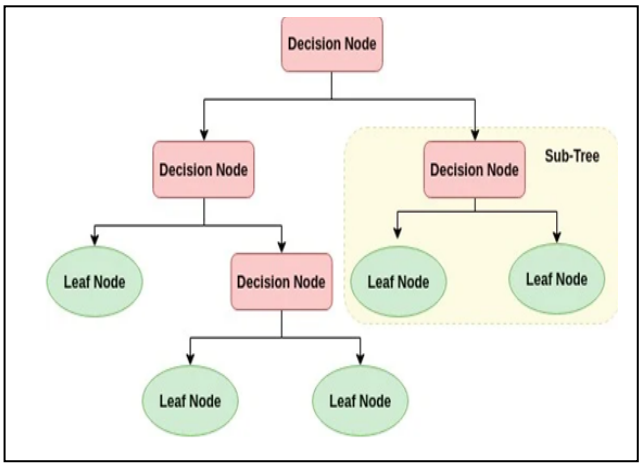

Fig. 6. Classification example of Decision Tree Algorithm

In Fig. 5. an Input test point is introduced, for which class needs to be inferred. Initially, the K-NN algorithm computes distances from this test data point to each point that exists in the dataset and then k neighbors with the minimum distances will be selected for class comparison. To calculate these distances, various approaches can be used for this purpose, but the most effective one is the Euclidean Distances formula (5). The cosine similarity function can also be used as a distance metric in the k-NN algorithm.

$$E(x, y) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2} \qquad (5)$$

3) Decision Tree: A decision tree is a flowchart-like tree structure in which an internal node represents a function (or attribute), a branch represents a decision law, and each leaf node represents the result. The root node is the topmost node in a decision tree. It learns to partition based on the value of an attribute. Recursive partitioning is a method of partitioning the tree recursively. This flowchart-like structure assists in making decisions. It is a flowchart diagram-style visualization that closely reflects human thought. Consequently, decision trees are simple to comprehend and interpret. The decision tree is a non-parametric or distribution-free approach that does not rely on probability distribution assumptions. With good accuracy, decision trees can handle high-dimensional data. The decision to make strategic splits has a significant impact on a tree's accuracy. To determine whether to divide a node into two or more sub-nodes, decision trees employ a variety of algorithms. The homogeneity of the resulting sub-nodes improves with the construction of sub-nodes. The decision tree divides the nodes into sub-nodes based on all available variables, then chooses the split that produces the most homogeneous sub-nodes [15].

To evaluate the further split, different parameters are used including Entropy, Information Gain, Gini impurity, etc, which will be explained in Random Forest (RF).

Fig. 6. depicts a classification example using the Decision Tree technique.

4) Random Forest (RF): Random Forest Classifier lies under the category of supervised classification algorithms to perform classification and regression functionalities. In the training phase, the RF algorithm creates a considerate amount of decision trees on the dataset and provides the label or class based on the calculation of the average prediction of each tree. Random forests are generated by combining three estimators in such a way that each tree which comprises of a random vector, is fitted on the dataset individually and with similar allocation for other trees available in the forest [16]. Although the RF algorithm is based on the decision tree idea, but RF model is proved to be more efficient in accurate inference which is achieved by expanding the multiple numbers of trees [17]. RF classifier is an ensemble classifier designed using various Decision tree models. The decision tree model is created using overall dataset samples including all elements of the dataset. While in the RF algorithm, only particular samples of dataset and features are randomly selected for training and classification purposes. As Decision tree algorithm is formulated on a greedy strategy that segregates the dataset into smaller chunks. The root node leads to the best-inferred value. At every decision branch, the data traits further are divided into two more branches and this process continues till the best-inferred variable is reached. To find out which node needs to be further divided, Entropy is calculated for each value by using (6) and then the difference between the parent and the child node is measured for each value to find out the Information gain [17].

$$Entropy = -p\ log\ 2\ (p) - q\ log\ 2\ (q) \qquad (6)$$

Gini is another well-known method implemented in the RF algorithm, which provides a measure of Impurities. The decision node, which evaluates the minimum Gini impurity is selected for further segregation. Gini impurity can be calculated using (7) [17].

$$Gini\ Impurity = 1 - \sum_{t=0}^{t=1} P_t^2 \qquad (7)$$

Where ***P*** is the probability of each class.

Fig. 7. illustrates a classification example using an RF algorithm. Each tree provides a probability assessment of the class label. The average Mean of these probabilities for 'n' trees is computed, and the maximum value produces the inferred class label. RF algorithm provides a reduction in the issue of Over-fitting, as it is working with average probabilities and rules out the biases.

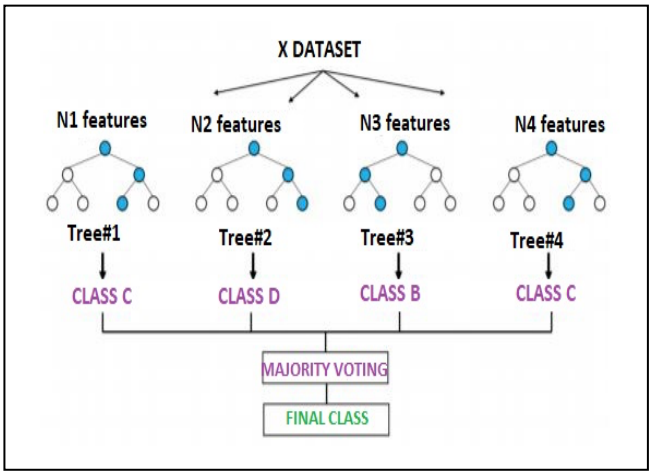

Fig. 7. Classification example of RF Algorithm

III. METHODOLOGY

As stated earlier, the main inspiration of this study is to explore and provide an ML-based solution for the discrimination of reflected sound signals occurring from two different objects. In this section, a complete workflow of the implemented approach for the realization of the Binary Classification model is presented. A step-by-step description of the applied model is provided. Furthermore, for the performance assessment of the trained classification models in the testing phase, some evaluation metrics are used, which will be explained later in this section.

To provide a convenient and simple usage of the proposed model, a graphical user interface (GUI) for the Binary Classification model is created. The implementation steps for the creation of GUI are also discussed in this section.

A. The workflow of Proposed Binary Classification Model

The block diagram of the proposed experimental setup to implement the Binary classification model is illustrated in Fig. 8.

1) Data Processing: To prepare the input time signal for the proposed classification model, some data processing steps are required. The training dataset is based on the comma-separated values (csv) files, which contain "one" time signal per row. These csv files, containing different time signals are allocated for each object i.e., Object#1 and Object#2. The data processing is performed on these csv files, as depicted in Fig. 9 and explained in the following steps.

a) CSV files merging: All the csv files, containing time signals for Object#1, are first merged into one csv file. This merged csv file represents all the time signals of Object#1 per row. In the same manner, the data of csv files containing time signals for Object#2 are also merged into a single csv file.

b) Conversion into excel file: The merged csv files for Object#1 and Object#2 are then converted into excel files (.xlsx format). These excel files for both objects serve as the input and are fed into the proposed experimental setup for further processing.

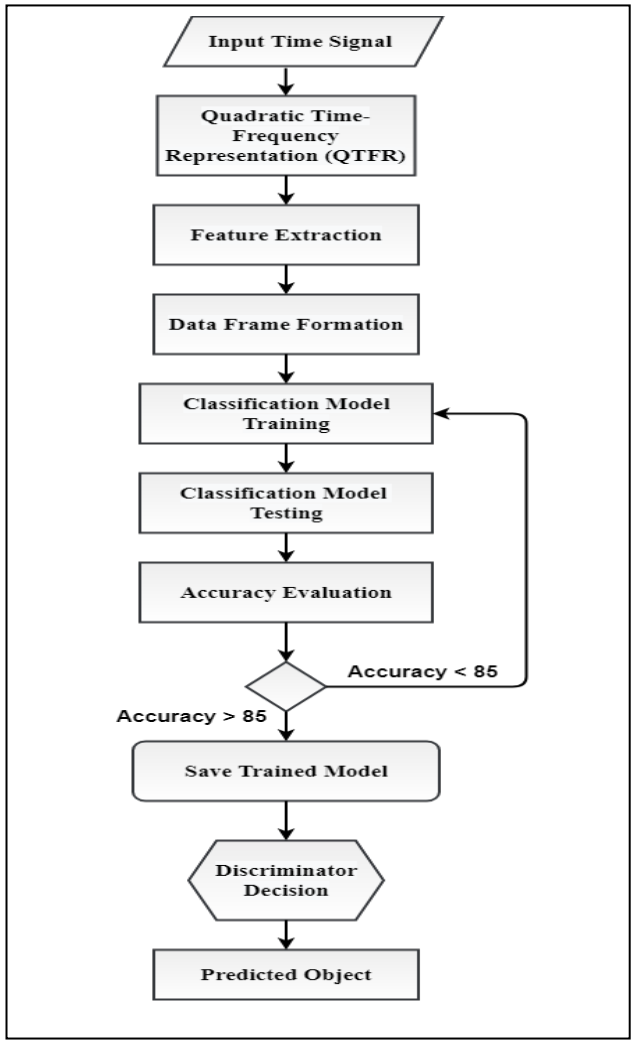

Fig. 8. The Workflow of the Binary Classification Model

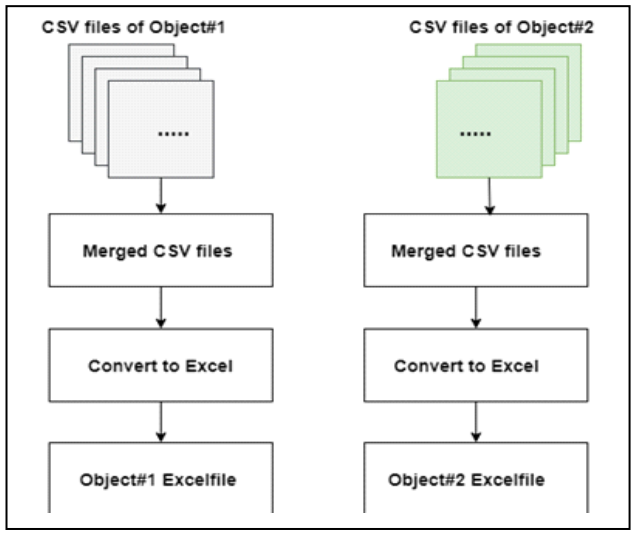

Fig. 9. Data Processing

2) Quadratic Time-Frequency Representation (QTFR): In this step, the system starts reading the given time signal one by one from the input excel files, and creates quadratic time-frequency representation for each time signal. As discussed earlier, the QTFR method is used to calculate the energy of a signal as a function of time and frequency. The resultant signal is termed as the quadratic time-frequency representation of the signal. Since the QTFR signal depicts the signal's energy density in the time-frequency domain, this can also be referred to as Spectrogram. There are different quadratic time-frequency signal analysis methods, out of which the Short-time Fourier transform (STFT) spectrogram method is chosen to implement in this experiment. Each one-dimensional input time signal is mapped into the two-dimensional time-frequency signal, which creates the spectrogram. These spectrograms are utilized in the feature extraction phase.

3) Feature Extraction: The feature extraction phase is the most vital element of any classification model and it serves as the first building block for designing and training of any ML-based classification model. The STFT spectrograms are subjected to the feature extraction phase. As specified in the previous section, six distinct features are extracted from the STFT spectrograms for training and testing of the model, which are as follows.

- Maximum Frequency of Spectrogram
- Spectrum Energy
- Instantaneous Power of Spectrogram
- Average Energy of Spectrogram
- Instantaneous Mean Frequency
- Group Delay

4) Data Frame Formation: In the feature extraction phase, the feature extraction process is applied on each STFT spectrogram, coming from both objects. To organize these extracted features, the data frame formation method is used. In this step, all the features that are extracted from the QTFR spectrograms of Object#1 are organized in a tabular format, as shown in Fig. 10. The target value "0" is assigned to these extracted features of Object#1.

Spectrogram Array	Max. Frequency	Spectrum Energy	Ins. Power	Average Energy	Ins. Mean Frequency	Group Delay	Target 0
[0]	MF[0]	SE[0]	IP[0]	AE[0]	IMF[0]	GD[0]	0
[1]	MF[1]	SE[1]	IP[1]	AE[1]	IMF[1]	GD[1]	0
[2]	MF[2]	SE[2]	IP[2]	AE[2]	IMF[2]	GD[2]	0
[3]	MF[3]	SE[3]	IP[3]	AE[3]	IMF[3]	GD[3]	0
...	...	...	...	...	...	...	...

*MF=Maximum Frequency *IP=Instantaneous Power *AE=Average Energy
*SE=Spectrum Energy *IMF=Instantaneous Mean Frequency *GD=Group Delay

Fig. 10. Dataframe Example for Object#1

In a similar manner, the extracted features from the QTFR spectrograms of Object#2 are subjected to Data formation, and target value "1" is assigned to them, as shown in Fig.11.

Spectrogram Array	Max. Frequency	Spectrum Energy	Ins. Power	Average Energy	Ins. Mean Frequency	Group Delay	Target 1
[324]	MF[324]	SE[324]	IP[324]	AE[324]	IMF[324]	GD[324]	1
[325]	MF[325]	SE[325]	IP[325]	AE[325]	IMF[325]	GD[325]	1
[326]	MF[326]	SE[326]	IP[326]	AE[326]	IMF[326]	GD[326]	1
...	...	...	...	...	...	...	1

*MF=Maximum Frequency *IP=Instantaneous Power *AE=Average Energy
*SE=Spectrum Energy *IMF=Instantaneous Mean Frequency *GD=Group Delay

Fig. 11. Dataframe Example for Object#2

5) Splitting Dataset for Testing and Training phase: At this step, the dataset is split up into "testing dataset" and "training dataset". The training dataset is used to train the classification model. The testing dataset is utilized to evaluate the final performance of the model. This testing dataset helps the model to learn for future predictions.

6) Binary Classification Model: In the previous step, the extracted feature-based training and testing datasets are obtained. In the classification phase, two operations are performed parallelly on classification models, namely "Model Training" and "Model Testing".

In the "Model Training" operation, the training dataset is utilized to train different classifiers for the realization of binary classification. The applied classification models are as follows.

- Logistic Regression
- k-Nearest Neighbors (k-NN)
- Decision Trees
- Random Forest (RF)

Each classifier trains itself with the given training data according to its algorithm as described in the previous section. Based on the distinct features and class labels of two objects, the classifiers learn the discrimination logic. As the binary classification is the main objective in this project, therefore the classifiers learn from the extracted features and associated labels that the input signal belongs to Object#1 or Object#2.

In the "Model Testing" operation, the testing data is fed into the classifiers to test whether the classifier's prediction is accurate or not. Based on accurate predictions, the efficiency of the classification model is evaluated. If the prediction accuracy of the classification model is greater than 85%, then the best model will be saved for future usage. If the evaluated accuracy is less than 85%, then it is assumed that the classification model needs to learn more, so this triggers model training operation again. The wrong predictions compensate to further update and train the classification model in order to achieve better efficiency.

Finally, the most effective and efficient ML classifier is chosen to implement the Binary classification model. To assess the performance of the conceived Binary classifier, some evaluation metrics are applied.

B. Performance Evaluation Metrics

The results predicted by the Binary classifier are evaluated using Confusion Matrix, which depicts the comprehensive performance of the Binary Classification model. In this 2-D matrix, rows and columns represent the Classes and the diagonal depicts the accurate classification, as shown in Fig. 12. The evaluation metrics highly depend on the values of True positive $\mathbf{t_p}$, True negative $\mathbf{t_n}$, False positive $\mathbf{f_p}$ & False negative $\mathbf{f_n}$. The $\mathbf{t_p}$ are the number of positive predicted tests, which are originally positive. Similarly, the $\mathbf{t_n}$ is the number of negative predicted tests, which are originally negative. The $\mathbf{f_p}$ is the number of positive predicted tests, which are originally negative and the $\mathbf{f_n}$ is the number of negative predicted tests that are originally positive [13].

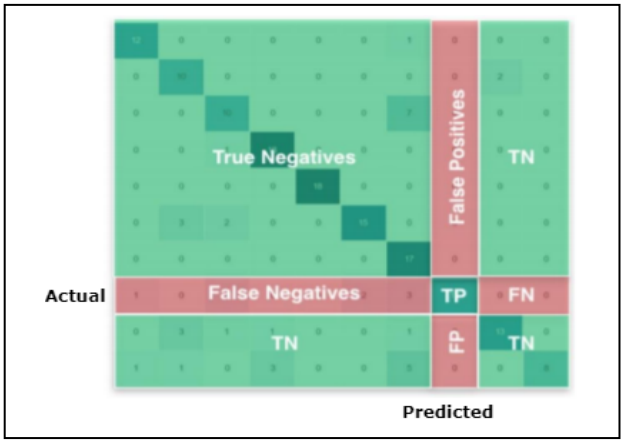

Fig. 12. Confusion Matrix [13]

The following measures, represented as (8), (9), (10), and (11), are used in the main evaluation metrics.

- False Discovery Rate (FDR)

$$FDR = \frac{f_p}{t_p+f_p} \tag{8}$$

- Negative Predictive Value (NPV)

$$NPV = \frac{t_n}{t_n+f_n} \tag{9}$$

- True Positive rate (TPR)

$$TPR = \frac{t_p}{t_p+f_n} \tag{10}$$

- True Negative Rate (TNR)

$$TNR = \frac{t_n}{t_n+f_p} \tag{11}$$

The evaluation metrics used in this project are described below.

1) Accuracy: Classification accuracy is defined as the percentage of the tests which are correctly predicted from total conducted tests. Accuracy can be evaluated by using [12].

$$Accuracy = \frac{t_p+t_n}{t_p+t_n+f_p+f_n} \tag{12}$$

2) Precision: This evaluation metric is defined as the ratio of the number of tests that actually have labeled class to the overall tests classified as a labeled class. Precision can be evaluated by using (13).

$$Precision = \frac{t_p}{t_p+f_p} \tag{13}$$

3) Recall: This evaluation metric is defined as the ratio of the number of tests that are classified as labeled class to overall tests which truly lie in labeled class. It is calculated by using (14).

$$Recall = \frac{t_p}{t_p+f_n} \tag{14}$$

4) F1 Score: This evaluation metric is defined as the ratio of the number of tests that are classified as labeled class to overall tests which truly lie in labeled class. It is calculated by using (15).

$$F1\ score = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{15}$$

5) ROC: A Receiver Operating Characteristic curve (ROC), is a graphical plot that depicts the prediction performance of a binary classification model when its "discrimination threshold" is changed. The ROC curve plots a parametric graph of the True Positive Rate of a threshold parameter versus the False Positive Rate of a threshold parameter, while the threshold serves as the variable parameter.

C. Graphical User Interface (GUI)

For the simple and convenient usage of the Binary classification model or Discriminator, a Graphical User Interface (GUI) application is created. The user just needs to enter a random csv/excel file from any Object's dataset, the desired starting row, and column values, and the saved ML model file. After computation, the GUI application will predict that the given file belongs to which object.

IV. Code Description

This section provides the step-by-step explanation of the implemented code for the realization of the Binary classification or Discriminator model and its GUI application. This ML-powered discriminator model is developed using Python.

A. Implementation of Binary Classification Model

Following is the detailed description of the implemented steps for the realization of the Binary classification model.

1) Data Processing: The first step is to convert the given dataset into the form which can be fed into the classification model. As stated earlier, the dataset for both Object#1 and Object#2 is given in the form of csv files. The following steps are performed on the provided dataset.

- The python-based program, dubbed "DataProcessing.py" runs first.
- The python modules used in this program are "glob" and "pandas".
- The folder path, which contains the data files (csv files) for Object#1 and Object#2, is provided.
- An array is created, which reads and appends all the csv files of Object#1.
- Similarly, an array is created for Object#2 files.
- To merge all of the csv files for Object#1, the function "pd.concat()" is called, which creates the "Object_1_Merged.csv" file. Similarly, the "Object_2_Merged.csv" file is created.
- For the conversion of merged csv files of Object#1 and Object#2 to excel (xlsx), "read_file.to_excel" method is used.
- The resultant files are "Object1.xlsx" and "Object2.xlsx".
- These files will be called by the main implementation program as the input dataset.

2) Reading Input Time Signal: The main implementation program "LatestImplementation.py" first reads the excel files of both objects, as depicted in Fig. 13. The excel file for Object#1, named "object1.xlsx" contains the time signals (reflected sound signal) arriving from Object#1. The excel file for Object#2, named "object2.xlsx" contains the time signals (reflected sound signal) arriving from Object#2.

```
obj_two_data = pd.read_excel('object2.xlsx', header = None)
obj_one_data = pd.read_excel('object1.xlsx', header = None)
```

Fig. 13. Reading the excel files of Object#1 & Object#2

3) Quadratic Time-Frequency Representation (QTFR) using STFT Spectrogram: In this step, the quadratic time-frequency representation of all the input time signals coming from both objects is created.

- The python libraries and modules used in this step include "matplotlib", "numpy", "chirp", "spectrogram", "stft", etc.
- The input arrays for Object#1 and Object#2 are created using function np.array().
- The method "Spectrum" is responsible to create the STFT spectrograms of the signal.
- The arguments including the array of Object#1, the number of data points used in each block for the FFT "NFFT", and sampling frequency "Fs" are passed in the function plt.specgram().
- This returns the QTFR or spectrogram of the given signal, as shown in the Fig. 14.
- Similarly, the QTFR or spectrograms of signals from Object#2 are created, as depicted in Fig. 15.

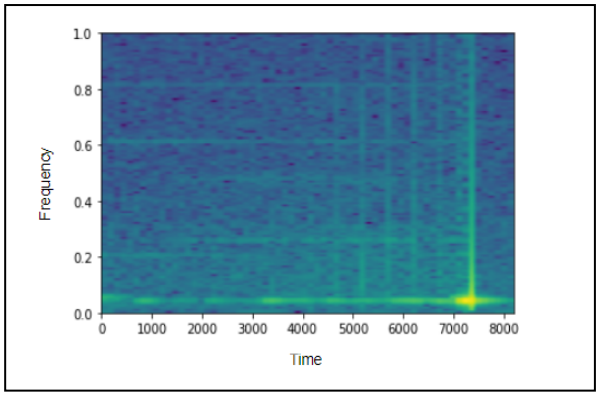

Fig. 14. STFT spectrogram of Object#1's signal

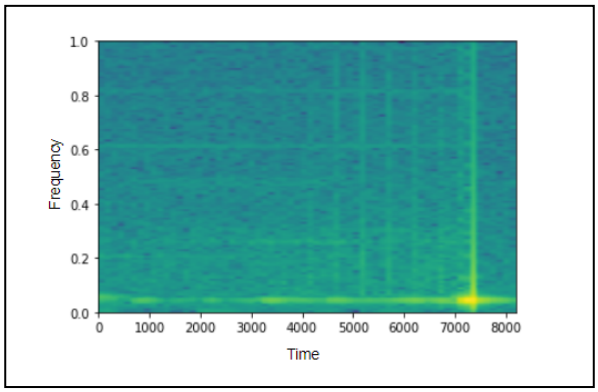

Fig. 15. STFT spectrogram of Object#2's signal

- These spectrograms of both objects are then subjected to the feature extraction phase in the next step.

4) Quadratic Time-Frequency Representation (QTFR) using WVD: For testing purposes, the Wigner Ville Distribution method is also examined in this project to obtain the QTFR of time signals of Object#1 and Object#2.

- The python module named "tftb.processing" is used for the implementation of the function WignerVilleDistribution().
- The obtained QTFR of one input signal of Object#1 is depicted in Fig. 16.

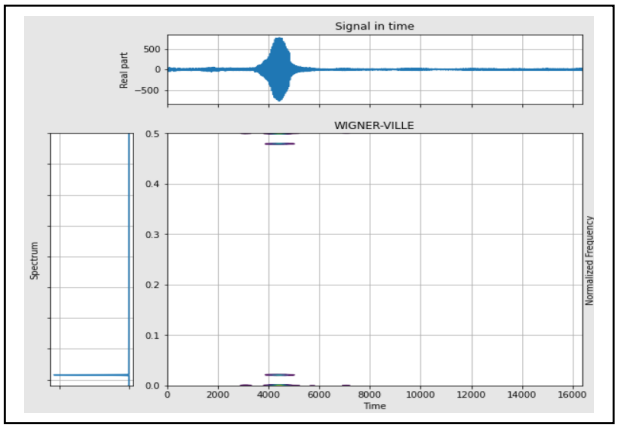

Fig. 16. WVD-based QTFR of Object#1's signal

- Similarly, the QTFR of one input signal of Object#2 is illustrated in Fig. 17.

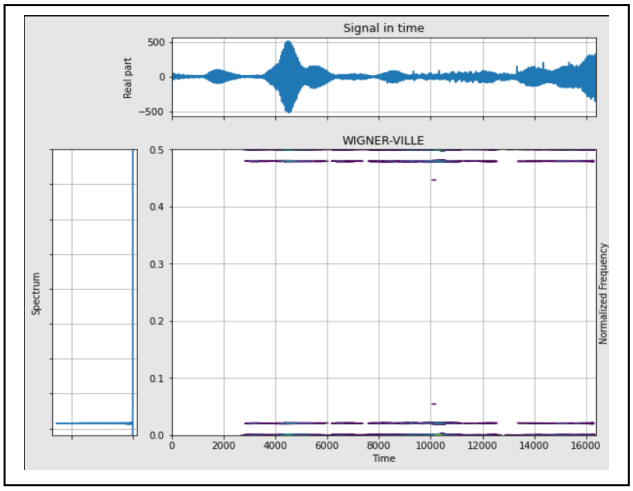

Fig. 17. WVD-based QTFR of Object#2's signal

- This function took approximately 5-10 minutes processing time for each time signal. When it was applied to the whole data set, the system crashed, due to which it did not proceed further.
- Due to the long processing time and memory requirements, the WVD-based QTFR is not applied in the final implementation of the project.

5) Feature Extraction Phase: As described in the previous section, six different types of features are extracted from the STFT-based spectrogram arrays of Object#1 and Object#2 using "numpy", as shown in Fig. 18.

```
instantaneous_power = (np.amax(abs(spectrum[0])))/t[np.argmax(abs(spectrum[0]))]
max_value = np.amax(abs(spectrum[0]))
sum_obj1.append(np.sum(spectrum))
max_values_obj1.append(max_value)
average_energy_obj1.append(np.sum(spectrum)/len(spectrum))
instantaneous_power_values_obj1.append(instantaneous_power)
instantaneous_frequency_array_object1.append(calculate_instantaneous_frequency(spectrum,freqs
TFA_Group_Delay_Object1.append(calculate_TFA_GroupDelay(spectrum, t))
```

Fig. 18. Methods for feature extraction

- The method "np.amax()" is used to compute the peak or "Maximum Frequency" from the spectrogram.
- The method "np.sum()" is used to compute "Spectrum Energy".
- The "Average Energy" is computed by dividing the "Spectrum Energy" with the length of the spectrum.
- The "Instantaneous Power of the Spectrum" is calculated by dividing the maximum value of the spectrum with the given time instant.
- The "Instantaneous Frequency" and " Group Delay" based features are computed by applying the formulas specified in (1) and (2).

6) Dataframe Formation: After extracting features from the QTFR spectrograms of both Objects, the data frames are created to organize the extracted features and to distinguish between the two classes.

- The data frames are created for organizing all six features extracted from the spectrograms of Object#1 using python module pandas function "pd.DataFrame()".
- For the extracted features of Object#1, the assigned labeled class is "Target=0".
- In a similar way, the data frame of Object#2 is created. The assigned labeled class for Object#2 is "Target=1".
- These data frames of extracted features of Object#1 and Object#2 are then merged into one data frame using the function "pd.concat", and the resultant data frame is shown in Fig. 19.

	MaxFrequency	Target	Spectrum Energy	Instantaneous Power	Average Energy	Instantaneous Frequency	TFA Group Dela
0	2.246051e+10	0	7.329066e+11	2.763350e+06	5.681446e+09	0.002625	42.50001
1	3.842053e+03	0	9.404116e+07	5.828358e-01	7.290013e+05	0.042834	186.38949
2	7.910337e+02	0	9.100440e+07	4.119967e+00	7.054604e+05	0.042971	186.82584
3	3.805260e+03	0	9.440205e+07	5.945719e+01	7.317989e+05	0.042503	184.74712
4	1.038515e+03	0	8.705725e+07	1.622680e+01	6.748624e+05	0.043077	187.74888
...	...	...	...	...	...	...	...
324	3.264063e+04	1	1.737546e+09	2.428618e+01	1.346935e+07	0.043026	194.62183
325	1.516015e+04	1	1.605577e+09	2.299780e+00	1.244634e+07	0.043140	192.25213
326	2.233882e+05	1	1.803108e+09	3.388777e+01	1.397758e+07	0.042592	195.17811
327	8.748207e+03	1	1.383324e+09	1.076305e+00	1.072344e+07	0.042356	184.94699
328	1.058132e+03	1	1.575284e+09	1.463125e-01	1.221151e+07	0.042508	188.17990

Fig. 19. Formated Data-frame

- These features-based data frames are used for the training of the binary classification model, which will be employed to distinguish between the signals of Object#1 and Object#2.

7) The Binary Classification Model Training: For the realization of the Binary Classification model or Discriminator, four different classifiers are trained and tested, namely Logistic Regression, k-Nearest Neighbors (k-NN), Decision Trees, and Random Forest (RF).

- The datasets are split into testing and training data. The datasets X_train and Y_train are used to train the models. While X_test and Y_test are used to test the model.
- The python library "sklearn" is used to implement these four ML classifiers. These classifiers are added in the function "model", as shown in Fig. 20.

```
models = {"Logistic Regression": LogisticRegression(),
          "KNN": KNeighborsClassifier(),
          "Random Forest": RandomForestClassifier(),
          "Decission Tree": tree.DecisionTreeClassifier()}
```

Fig. 20. Methods used to apply ML classifiers

- For the cross-validation, the function "fit_and_score", shown in Fig. 21, is created to train and evaluate the ML models based on accurate prediction.
- In this function, a loop is initiated to fit the models to the training datasets. All four ML classification models will learn as per the provided training dataset. The testing dataset is also fed into this loop, which will check each model's performance based on accurate predictions. This function returns the model

scores, which will be kept in a dictionary. A performance threshold is set for judging the efficient classifier. If the accuracy is greater than 85%, then the model will be saved, otherwise, it will continue learning by train and test datasets.

```
def fit_and_score(models, X_train, X_test, y_train, y_test):

    # set a random seed
    np.random.seed(42)
    #Make a dictionary to keep model scores
    model_scores = {}
    #Loop through models
    for name, model in models.items():
        # fit the model to the data
        model.fit(X_train, y_train)
        # evaluate the model and append its score to model_scores
        model_scores[name] = model.score(X_test, y_test)
    return model_scores
```

Fig. 21. Fit & Score function

- In this function, a loop is initiated to fit the models to the training datasets. All four ML classification models will learn as per the provided training dataset. The testing dataset is also fed into this loop, which will check each model's performance based on accurate predictions. This function returns the model scores, which will be kept in a dictionary. A performance threshold is set for judging the efficient classifier. If the accuracy is greater than 85%, then the model will be saved, otherwise, it will continue learning by train and test datasets.
- To save the best-trained model for binary classification, a python module "Pickle" is used. Since, the RF classifier turns out the best one with the highest accuracy, the trained model using RF classifier is saved for the realization of the Binary classification model.

8) The Binary Classification Model Testing: To test the effectiveness of the conceived Binary classification model or Discriminator with RF classifier, the following steps are performed.

- A random test file is given as input to the system.
- The already trained classification model extracts the features of the given input and performs feature matching.
- Based on feature matching, the model predicts if the given test signal belongs to Object#1 or Object#2.
- The effectiveness of the predicted results then judged by the performance evaluation metrics.

B. *Performance Evaluation Metrics*

To quantify the performance of the conceived Binary classification model, some performance measuring steps are applied. For this purpose, the python tool "scikit.learn" is used.

1) Confusion Matrix: A 2-D matrix, which is used for the performance evaluation of the Binary classification model.

- With the sklearn.metrics and seaborn modules, a confusion matrix is created by using the actual values versus the predicted values, as shown in Fig. 22.

```
# Confusion Matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
y_actual = y_test
y_est = models["Random Forest"].predict(X_test)
cm = confusion_matrix(y_actual, y_est).ravel()
tn, fp, fn, tp  = cm
disp = ConfusionMatrixDisplay(confusion_matrix=cm.reshape(2,2))
disp.plot()
```

Fig. 22. Obtaining performance measures from Confusion Matrix

- The values of True positive t_p, True negative t_n, False positive f_p & False negative f_n are derived from the confusion matrix.

2) Computing FDR, NPV, TPR, TNR: These classification assessment measures are computed with the derived values t_p, t_n, f_p and f_n by using the formulas specified in (8), (9), (10), and (11).

3) Other evaluated metrics: Following are the performance evaluation metrics that are derived from the above-computed measures. By using the sklearn.metrics module.

- Accuracy
- Recall
- Precision
- F1 score
- ROC plot

C. *Graphical User Interface (GUI)*

The GUI application is implemented using python libraries and modules including tkinter, pandas, numpy matplotlib.pyplot, etc.

Following are the implementation steps for GUI.

- The GUI application can be run with the python program named "GUI_BinaryClassifier.py". This program contains the class "Binary Classifier" which consists of six functions.
- The first function, named "__init__()", uses the python library tkinter to design the outlook of the GUI.
- This function adds and organizes the buttons and fields such as the "Browse" button to choose the input test file and to choose the trained model, the fields "Enter Row number", "Enter starting column", "Enter Signal Length", buttons "Predict Object", "Show model assessment", "Try again" and "Exit". The progress bars are also included in this function.
- To make the GUI user-friendly, the step-by-step execution is also highlighted in the GUI application.
- In the first step, the button "Browse" calls the second function, dubbed "openfile()" which imports the csv file path.
- While the chosen file is being loaded to the application, a progress bar indicates the processing progress.

- In step 2, the user can enter the values for fields “Enter Row number”, “Enter starting column” and “Enter Signal Length”, to obtain the desired input signal for prediction.
- In step 3, the button “Browse” to choose a trained model calls the third function, named “choosemodel()”, where the already trained and saved RF classifier-based Discriminator model will be loaded.
- Here, the progress bar is added again to indicate the processing progress
- In step 4, the button “Predict Object” calls the function termed “predict()”, which performs all the steps that are specified in the main program to predict the Object associated with the input signal.
- The results are then depicted in the field “Predicted Object” and “# of Scanned points” below the Predict button.
- In addition to that, the button “Show Model Assessment” calls the function “popupmsg()”, which pops up another window that contains all the performance assessment metrics values of the chosen ML classifier model.
- The button “Try Again?” calls the function “reset()” which clears all the fields of the GUI to enable the user to test another file.
- The button “Exit” simply exits the GUI application.
- The title assigned to the GUI window is “Discrimination of Reflected Sound signal”.
- The preview of this GUI application is illustrated in Fig. 23.

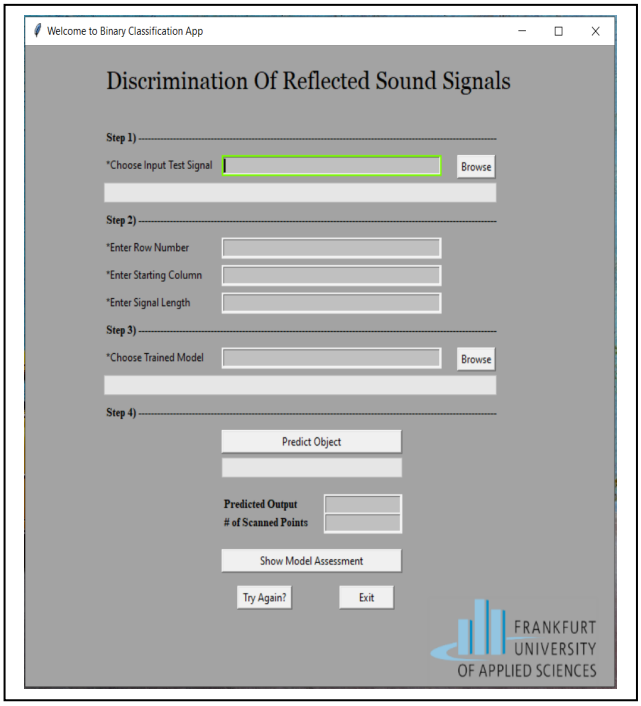

Fig. 23. The GUI preview

V. Results

The motivation of this project is to implement an efficient Binary classification model which could discriminate between the sound signals reflecting from two distinct objects. The achieved results and outcomes in the realization of such a Discriminator are discussed below.

A. The Implemented Binary Classification Model

To achieve this goal, the Binary classification or Discriminator model is realized. Four different ML classification models Logistic Regression, k-Nearest Neighbors (k-NN), Decision Trees, and Random Forest (RF) are considered and examined for the implementation of the Discriminator. After training these ML classifiers according to the QTFR-based extracted features, the most efficient ML classification model is chosen. As depicted in Fig. 24, the RF classifier stands out among all the other classifiers with 95.7%. Accuracy, which is the highest. Consequently, the ML model trained with the RF classifier is stored for future utilization as the Binary Classification Model.

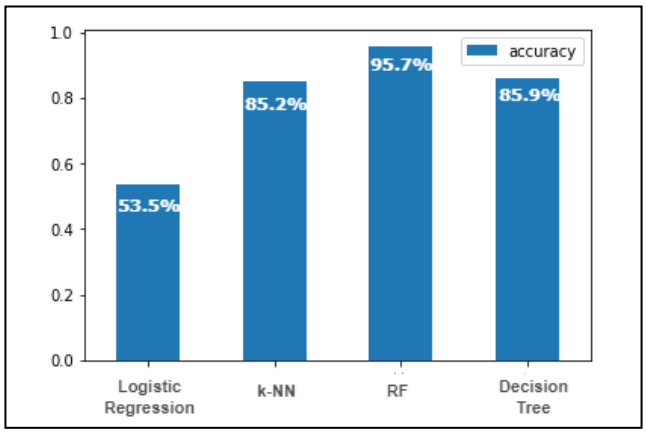

Fig. 24. Assessment of best ML classifier

B. Model Performance Evaluation

The main agenda is that the implemented Binary classification model must predict the correct Object by extracting and matching features of the provided reflected signal. To assess the implemented Binary classification or Discriminator model using the RF classifier, Some evaluation metrics are applied. Following are the results of these metrics.

1) Confusion Matrix: The obtained Confusion matrix is depicted in Fig. 25. The values of True positive t_p, True negative t_n, False positive f_p & False negative f_n are derived from this confusion matrix.

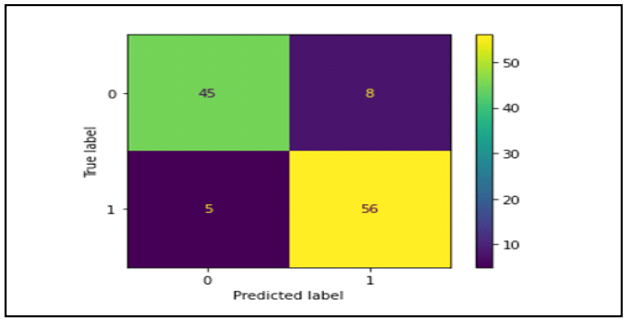

Fig. 25. Obtained Confusion matrix

2) Computing FDR, NPV, TPR, TNR: These values are computed with the derived values t_p, t_n, f_p and f_n, as shown in Fig. 26.

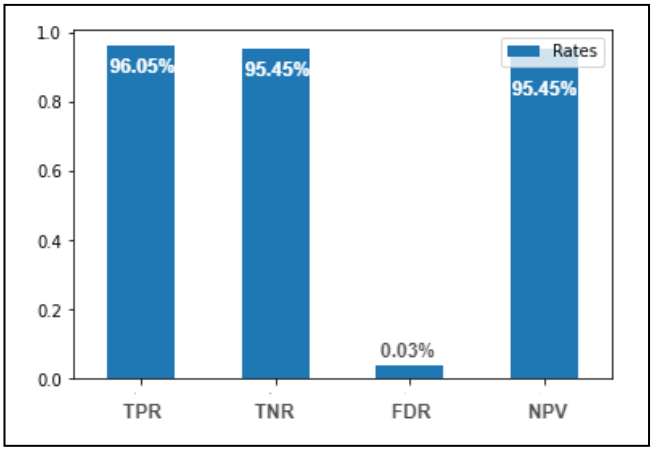

Fig. 26. Evaluated Assessment metrics

3) Other Evaluation Metrics: The performance assessment of the implemented Binary Classifier can also be performed by computing metrics such as Accuracy, Precision, Recall, and F1 score. The evaluated metrics are presented in Table I.

TABLE I. RESULTS OF EVALUATION METRICS

Evaluation Metric in %	Binary Classification Models
Accuracy	95.77%
Precision	96.05%
Recall	96.05%
F1 score	96.05%

4) ROC Plot: The obtained ROC plot is depicted in Fig. 27.

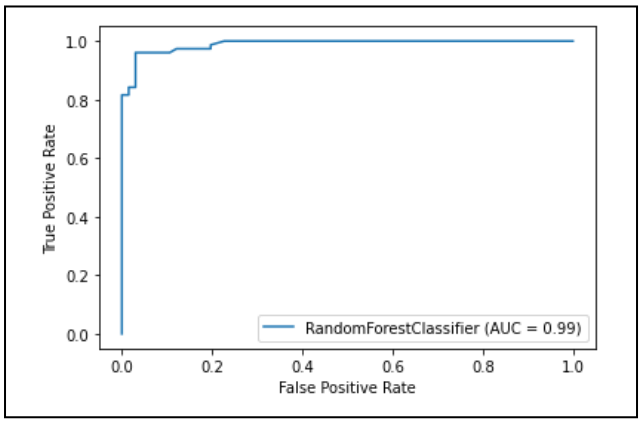

Fig. 27. ROC Plot

The evaluated measures and metrics depict the efficient performance of the realized Binary classification model with an accuracy of 95.77%.

C. Discriminator's Predicted Results via GUI

To check the performance of the created GUI application for Binary classification, a random csv file from the dataset of Object#1 is provided to the application, as shown in Fig. 28. After loading the Discriminator model, it predicted that the given signal belongs to Object#1. Fig. 29. proves the accurate prediction by GUI application.

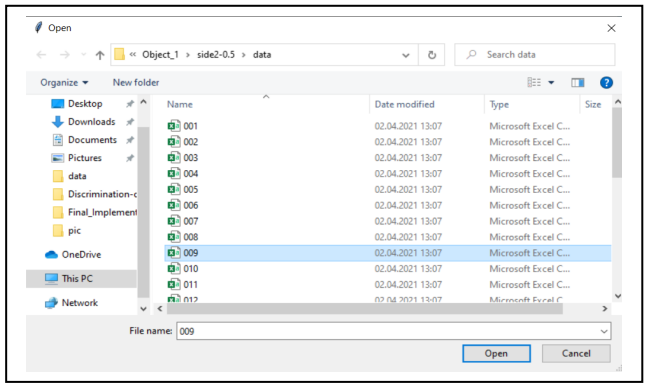

Fig. 28. Selected csv file from Object#1's dataset

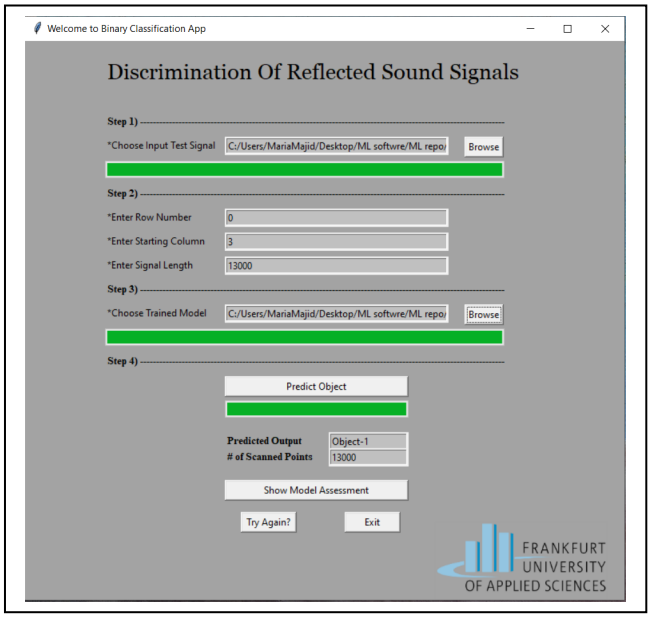

Fig. 29. Predicted Result by GUI-based Discriminator

The obtained STFT spectrogram of the input test signal is represented in Fig. 30.

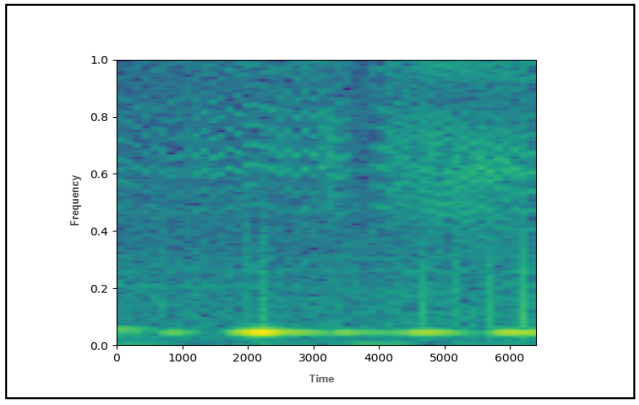

Fig. 30. STFT spectrogram of test signal (Object#1)

When the "Show Model Assessment" button is pressed, it pops up another window that depicts the values of model performance evaluation measures, as shown in Fig. 31.

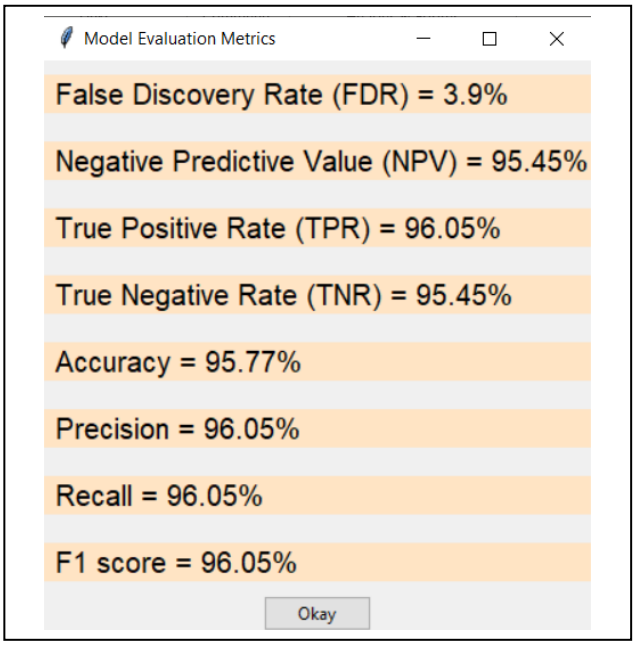

Fig. 31. The obtained Model Assessment measures

VI. Guidelines to Rebuild the Proposed Solution

The steps to rebuild the proposed solution are listed as follows.

- The pre-requisites include the installation of Python3.8 and pip.
- Some Python packages and modules need to be installed prior to running the main program. These packages include glob, pandas, numpy, scikit-learn, sklearn.utils,sklearn.metrics,sklearn.model_selection ,sklearn.linear_model,sklearn.neighbors,sklearn.ensemble, pickle, tkinter, matplotlib.pyplot, tkinter.font, xlrd, and openpyxl. To install these packages, the following command will be used.

Pip install <package name>

- Since python is a scripting language, there is no need to build the program. The programs can be run by simply using the python3 command.
- Before running the program "DataProcessing.py", change the folder path to the current path where the dataset files are located.
- Firstly, the python program named "DataProcessing.py" will be executed by using the command "python3 DataProcessing.py".
- Then the main program "LatestImplementation.py" will be executed next by using "python3 LatestImplementation.py".
- To run the GUI application, execute the python program by using "python3 GUI_BinaryClassifier.py".

VII. Conclusion

In this report, a comprehensive experimental setup of an efficient Binary classification model or Discriminator using Machine learning algorithms is presented. To design the framework, the creation of Quadratic Time-Frequency representation (QTFR) of signal, and different ML classification algorithms are discussed. To obtain the QTFR based spectrogram of time signals, the STFT spectrogram method is used. For distinct feature extraction from QTFR-based spectrograms, the features such as "Maximum Frequency", "Spectrum Energy", Instantaneous Power", "Average Energy", "Instantaneous Mean Frequency", and "Group Delay" work impressively. For the Binary classification model training and testing tasks, four machine learning classification algorithms Logistic Regression, k-Nearest Neighbors (k-NN), Decision Trees, and Random Forest (RF) are examined and assessed from which the performance of the RF classification model stand out compared to the other three classifiers. Hence, the Binary classification model or Discriminator is implemented using the trained RF classifier model. For ease of usage, a GUI application is provided which loads the trained Binary classification model and predicts the Object results after processing the given signal.

Individual Contributions

Name of Group member	Contributions in project
Chetan Chadha	* Implemented method to perform Data Rendering * Did Research and implemented WVD based spectrogram * Implemented training and predicting methods in the main class * Created Dataframes and labeled them with different Target values * Implemented a method that can train 4 different ML models * Generated Confusion Matrix and other evaluation metrics * Enhanced GUI in terms of time and accuracy * Documented techniques using WVD *Documented Section IV "Code Description" in the Report * Also contributed in writing Results discussion
Maria Majid	* Performed Data processing to merge csv files into 1 excel file * Created and Improved the GUI Interface by using multiple Tkinter libraries * Implemented train and prediction method in GUI * Added step-by-step documentation in the code * Did Research and implemented STFT spectrogram in the main class * Implemented methods for feature extraction in the main implementation *Documented section III "Methodology" in project report *Documented code description for GUI application *Documented Guidelines on how to build the project in the report *Also Contributed to Results discussion

Name of Group member	Contributions in project
Sanjana Sharma	*Added Abstract and Introduction in the project Report *Documented ML classifier theory in report *Added code In-line documentation *Performed testing on GUI with multiple scenarios *Also contributed in result discussion *Did initial research on how to create GUI *Implemented file explorer functionality to upload the files in GUI

ACKNOWLEDGMENT

We would like to thank Prof. Dr. Andreas Pech for providing us the opportunity to work on this topic under his supervision and for providing us the proper guidance to complete the assigned project and this report.

REFERENCES

[1] G. Papakostas, K. Diamantaras and F. Palmieri, “Emerging Trends in Machine Learning for Signal Processing”, Computational Intelligence and Neuroscience, vol 2017, pp. 1-2, November 2017, Article ID 6521367, DOI: 10.1155/2017/6521367

[2] M. J. Bianco, P. Gerstoft, J. Traer, E. Ozanich, M. A. Roch, S. Gannot, and C. A. Deledalle, "Machine learning in acoustics: Theory and applications", The Journal of the Acoustical Society of America 146, pp. 3590-3628, 2019, DOI: 0.1121/1.5133944

[3] A. C. de Carvalho and A. A. Freitas, "A tutorial on multi-label classification techniques," in Foundations of Computational Intelligence, vol. 5, ed: Springer, pp. 177-195, 2009

[4] M. Er, R. Venkatesan, & N. Wang, “An Online Universal Classifier for Binary, Multi-class and Multi-label Classification”, September 2016.

[5] F. Hlawatsch, A. Papandreou-Suppappola, and G. Boudreaux-Bartels, “The power classes-quadratic time-frequency representations with scale covariance and dispersive time-shift covariance”, IEEE Transactions on Signal Processing, 47(11), 1999, pp. 3067–3083. DOI: https://doi.org/10.1109/78.796440)

[6] The National Instrument Corporation, “Understanding Quadratic Time Frequency Analysis Methods (Advanced Signal Processing Toolkit)”, LabVIEW 2010 Advanced Signal Processing Toolkit Help, June 2010, Available on: https://zone.ni.com/reference/en-XX/help/371419D-01/lvasptconcepts/tfa_quadratic/

[7] F. Auger, P. Flandrin, P. Gonçalvès, and O. Lemoine, “Time-Frequency Toolbox, for use with MATLAB”, 1995-1996. Available on: http://tftb.nongnu.org/tutorial.pdf.

[8] L. Cohen, “Time-Frequency Analysis: Theory and Applications”, Chapter 8, Prentice Hall, 1994.

[9] The National Instrument Corporation, “Spectrogram Feature Extraction (Advanced Signal Processing Toolkit)”, LabVIEW 2010 Advanced Signal Processing Toolkit Help, June 2010, Available on: https://zone.ni.com/reference/en-XX/help/371419D-01/lvasptconcepts/tfa_spectrogram_features/#Mean_Instantaneous_Frequency

[10] C. Xu, C. Wang, & J. Gao, “Instantaneous Frequency Identification Using Adaptive Linear Chirplet Transform and Matching Pursuit”, Shock and Vibration, 2016, pp. 1-10. DOI:10.1155/2016/1762010.

[11] J. Brownlee, “4 Types of Classification Tasks in Machine Learning”, Machine Learning Mastery, August 2020. Available on: https://machinelearningmastery.com/types-of-classification-in-machine-learning/

[12] J. Brownlee, “Logistic Regression for Machine Learning”, Machine Learning Mastery, August 2020, Available on: https://machinelearningmastery.com/logistic-regression-for-machine-learning/

[13] A. M. Sharma, "Speaker Recognition Using Machine Learning Techniques", Master's Projects, SJSU Scholarworks, 685. DOI: https://doi.org/10.31979/etd.fhhr-49pm https://scholarworks.sjsu.edu/etd_projects/685

[14] C. Liu, “A Top Machine Learning Algorithms explained: Support Vector Machines”, Feb 2020. Available on: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

[15] A. Navlani, “Decision Tree Classification in Python”, Datacamp com, December 2018, Available on: https://www.datacamp.com/community/tutorials/decision-tree-classification-python

[16] L. Breiman, “Random Forests, “Machine learning”, vol. 45, no.1, pp.5-32, 2001.

[17] K. Kirasich, T. Smith and B. Sadler, “Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets”, SMU Data Science Review, vol. 1, no. 3, article 9, 2018.