Improvisations in Actor Model on top of   
Service Bus

Master of Engineering in Information Technology  
Individual Project  
Chetan Chadha 1324510  
chetan.chadha@stud.fra-uas.de

*Abstract*— ………...

Keywords— ………..

# Introduction

Today’s computing environments are vastly different from the ones in use when the programming models of yesterday were conceived. Actors were invented decades ago but relatively recently their applicability to the challenges to the modern computational systems has been recognized and proved to be effective. The actor model allows you to think about your code in terms of communication. An Actor is a fundamental unit of computation which allows you to perform basic functions such as create another actor, send a message and designate how to handle the next message. Actors are light weight and easy to create hundreds and millions of them. Moreover, fewer resources are consumed when compared with threads. Actors are isolated from each other and do not share memory. They have a state and the only way to change it is by receiving a message. Actors can communicate with each other only with the help of message. A message can be described as a simple, immutable data structure than can easily send over the network. Only one message can be processed by an actor at one time. Actors work asynchronously that means they do not need to wait for a response from another actor. Actors have a adress

Distributed computing is a model in which the components of a software system are shared among multiple computers. These components are spread out across multiple computers, they run as one system. This enhances the efficiency and performance.

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 paper presents an ML-powered solution for the discrimination of reflected sound signals by using a 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 and 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. In Section V, the result analysis and assessment for the implemented binary classification model based on some evaluation metrics will be discussed. Finally, the paper is concluded in Section IV.

# Techniques used in Binary Classification Model Implementation

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

## Quadratic Time-Frequency Representation

QTF and STFT algorithm and working tbd.

## Machine Learning Classification Models

Description of ML Models for binary classification tbd.

# Methodology

As stated earlier, the main inspiration of this study is to 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.

## 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 x.

Diagram

Description automatically generated

### Data Processing: The training dataset is based on the comma-separated values (csv) files, which contain a 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 figure x.

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

#### 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.

Diagram

Description automatically generated

### 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.

### 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. Two distinct features are extracted from the STFT spectrograms for training and testing of the model, which are as follows.

* Maximum Frequency (the maximum frequency of the spectrogram of one time signal)
* Maximum Spectrum Sum (the sum of all the elements in the spectrogram of one time signal)

### Data Frame Formation: In the feature extraction phase, the feature extraction process is applied on each time-frequency spectrogram, coming from both objects. To organize these extracted features, the data frame formation method is used. In this step, all the “Maximum Frequency” & “Maximum Spectrum sum” features extracted from the QTFR spectrograms of Object#1 are organized in a tabular format, as shown in Table I. The target value “0” is assigned to these extracted features of Object#1.

1. Data Frame Formation of Object#1

| Spectrogram Array | Max Frequency | Max Spectrum Sum | Target |
| --- | --- | --- | --- |
| 0 | Max Frequency0 | Max Spectrum Sum0 | 0 |
| 1 | Max Frequency1 | Max Spectrum Sum1 | 0 |
| … | … | … | 0 |

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 Table II.

1. Data Frame Formation of Object#2

| Spectrogram Array | Max Frequency | Max Spectrum Sum | Target |
| --- | --- | --- | --- |
| 324 | Max Frequency | Max Spectrum Sum | 1 |
| 325 | Max Frequency | Max Spectrum Sum | 1 |
| … | … | … | 1 |

### 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.

### 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 goal in this study, 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 are used 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.

## 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 x. The evaluation metrics highly depend on the values of True positive **tp**, True negative **tn**, False positive **fp** & False negative **fn**. The **tp** are the number of positive predicted tests, which are originally positive. Similarly, the **tn** is the number of negative predicted tests, which are originally negative. The **fp** is the number of positive predicted tests, which are originally negative and the **fn** is the number of negative predicted tests that are originally positive.

The following measures are used in the main evaluation metrics.

* False Discovery Rate (FDR)
* Negative Predictive Value (NPV)
* True Positive rate (TPR)
* True Negative Rate (TNR)

The evaluation metrics used in this project are described below.

### 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 eq.

### 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 eq.

### 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 eq.

### 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 eq.

### ROC: A receiver operating characteristic curve (ROC) curve, 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 False Positive Rate of a threshold parameter, while the threshold serves as the variable parameter.

## 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 has to enter a random csv file from any Object’s dataset and the saved ML model. After computation, the system will predict that the given file belongs to which object.

# 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 model is developed using Python.

## Implementation of Binary Classification Model

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

### 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 used, 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.

### Reading Input Time Signal: The main implementation program “Implementation.py” first reads the excel files of both objects. 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.



### Quadratic Time-Frequency Representation (QTFR): In this step, the quadratic time-frequency representation of all the input time signals coming from both objects are 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. x.

Shape

Description automatically generated

* Similarly, the QTFR or spectrograms of signals from Object#2 are created, as depicted in fig x.

A picture containing chart

Description automatically generated

* These spectrograms of both objects are then subjected to the feature extraction phase.

### Feature Extraction Phase: From the QTFR-based spectrograms of Object#1 and Object#2, two types of features are extracted using “numpy”, as shown in fig. x.

Text

Description automatically generated with medium confidence

* The method “np.amax()” is used to compute the absolute “Maximum Frequency” from the spectrogram.
* The method “np.sum()” is used to compute “Maximum Spectrum Sum”.

### 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.

* A data frame is created for organizing the “Maximum frequency” features of all the spectrograms of Object#1 using “pd.DataFrame(max values\_obj1, columns=["MaxFrequency"])”.
* Similarly, another data frame is created for the “Maximum Spectrum Sum” features of all the spectrograms of Object#1 using “pd.DataFrame(sum obj1, columns=["MaxSpectrumSum"])”.
* 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”, as shown in fig x.
* These features-based data frames are used for the training of the binary classification model, which will be used to distinguish between the signals of Object#1 and Object#2.

Table

Description automatically generated

### The Binary Classification Model Training: For the realization of the Binary Classification model or Discriminator, four different classifiers are used, 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 the fig.

Text

Description automatically generated

* For the cross-validation, the function “fit\_and\_score”, shown in fig x, is created to train and evaluate the ML models based on accurate prediction.

Text

Description automatically generated

* 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 future use.

### 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.

## 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.

### 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.

Text

Description automatically generated

* The values of True positive , True negative , False positive & False negative are derived from the confusion matrix.

### Computing FDR, NPV, TPR, TNR: These classification assessment measures are computed with the derived values , , and by using the formulas specified in the eq. () () () ().

### 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

## 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 four functions
* The first function, named “\_\_init\_\_()”, uses the python library tkinter to design the outlook of the GUI. This adds and organizes the buttons such as “Choose a file”, “Choose a model”, “Predict” and “Exit”.
* The button “Choose a file” calls the second function, dubbed “openfile()” which imports the csv file path.
* The button “Choose a model” calls the third function, named “choosemodel()”, where the already trained and saved RF classifier-based Discriminator model will be loaded.
* The button “Predict” calls the fourth function termed “predict()”, which performs all the steps that are specified in the main program “Implementation.py” to predict the Object associated with the input signal. The results are then depicted in the field adjacent to the Predict button.
* The title assigned to the GUI window is “Sound signal discrimination”.
* The preview of this GUI application is illustrated in Fig.x.

Graphical user interface, application

Description automatically generated

# 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.

### 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. x, the RF classifier stands out among all the other classifiers with 88.5%. Accuracy, which is the highest. Consequently, the ML model trained with the RF classifier is stored for future utilization as the Binary Classification Model.

Chart, bar chart

Description automatically generated

### 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.

#### Confusion Matrix: The obtained Confusion matrix is depicted in Fig.x. The values of True positive , True negative , False positive & False negative are derived from this confusion matrix.

Chart

Description automatically generated

1. Example of a figure caption. (*figure caption*)

#### Computing FDR, NPV, TPR, TNR: These values are computed with the derived values , , and , as shown in the Fig. x.

Chart, bar chart

Description automatically generated

#### 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 III.

1. Results of Evaluation Metrics

| Evaluation Metric in % | Binary Classification Models |
| --- | --- |
| **Accuracy** | 87.5% |
| **Precision** | 87.5% |
| **Recall** |  |
| **F1 score** |  |

#### ROC Plot: The obtained ROC plot is depicted in Fig. x.

Chart, histogram

Description automatically generated

The evaluated measures and metrics depict the efficient performance of the realized Binary classification model with an accuracy of 87.5%.

### 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.x. After loading the Discriminator model, it predicted that the given signal belongs to Object#1. Fig. x. proves the accurate prediction by GUI application.

Graphical user interface, table

Description automatically generated

Graphical user interface, application

Description automatically generated

# 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. For distinct feature extraction from QTFR-based spectrograms, the features such as maximum frequency and sum of Spectrum 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. 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.

##### 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](https://www.researchgate.net/deref/http%3A%2F%2Fdx.doi.org%2F10.1155%2F2017%2F6521367?_sg%5B0%5D=XWGBoldQewsJBnvG7YxetYkUFg7uR1V_B2pA9ndw1fmCp5l9hf_N_L2AHjlGfLtWdy2andutwgqn1ROjRCw9Vl3FKQ.iq11NVTyIfEmKHmDS6lxbaTc2yhelQsJoEkQ_XFhShVW0TLLugDF7CPeRBu3IGgHjj2L01xBrET0e0a1g2SvAQ)
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. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
6. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
7. K. Elissa, “Title of paper if known,” unpublished.
8. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
9. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
10. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.