Discrimination of Reflected Sound Signals using Multi-Layer Perceptron (MLP)

Master of Engineering
Information Technology
Deepali Ashok Betkar
1324536
deepali.betkar@stud.fra-uas.de

Master of Engineering
Information Technology
Manash Chakraborty
1325085
manash.chakraborty @stud.fra-uas.de

Master of Engineering
Information Technology
Raghavendra Yarlagadda
1324413
raghavendra.yarlagadda @stud.fra-uas.de

***Abstract*— A sound wave upon its transmission doesn't just stop after reaching the end of the medium. Certain behaviors will be exhibited by the sound wave such as reflection off the obstacle, diffraction around the obstacle, and transmission through the obstacle. By recording the reflected signal, a time signal can be generated. This time signal results from the convolution of the incident acoustic wave with the surface properties of the reflecting object. By analyzing the reflected signal one can conclude on the reflecting object. The Machine learning (ML) techniques have enabled significant advancements in automated data processing and pattern recognition capabilities in a variety of fields. ML in acoustics is a promising development with many credible solutions to the acoustics challenges. This paper reviews the ML model called MLP (Multi-Layer Perceptron) to discriminate the reflected sound signal using the time signal and to predict the type of reflecting object.**

Keywords*— *Sound Waves, Reflection, Machine Learning, MLP, Accuracy, Classification, Prediction, Confusion Matrix, Measurements

I. INTRODUCTION

Sound transiting through air travels in a circular path from the source outwards. The majority of the sound would be echoed if it hits a particular medium, such as a wall, a vehicle, or even a human. The leftover sound would be absorbed by the wall. The sound reflects most of the sound back to the recipient depending on the material and shape of the piece, which makes reflections a significant part of the recipient's sound. The theory that "angle of incidence equals angle of reflection," also known as the law of reflection, governs sound reflection. A billiard ball bounced off the side of a table has the same effect as light and waves. The reflected time signal results from the convolution of the incident acoustic wave with the surface properties of the reflecting object. [1]

Automatic Object Recognition (AOR): Automatic object recognition is the capacity of an algorithm or computer to identify targets or other objects based on data collected from sensors. The object detection was initially achieved by the use of an audible image of the obtained signal, which was decrypted by an operator trained to classify radar light targets. While these skilled operators were effective, automatic classification methods have been introduced and are still being developed, allowing for greater precision and speed. AOR can also be used to classify man-made entities like cars and planes, as well as biological targets like wildlife species, humans.

A. Why object recognition is important?

Automatic object or target recognition is suitable for a number of tasks, including locating an object on the battlefield and removing disturbances caused by a large flock of birds on Doppler weather radar. Object recognition is a critical technology for military operations that has yet to realize its full tactical potential. A fully reliable recognition system can improve the warfighter's and platform's efficacy and survivability. The system uses sensor data to process information for decision making. Rapid acquisition and servicing of targets improves the weapon platform's and soldier's lethality and survivability. [2]

B. Approach to recognize an object

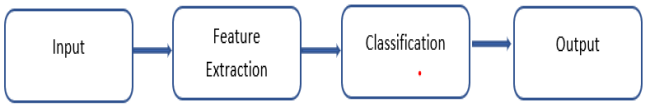

Figure 1 Machine Learning Approach

Many classical signal processing techniques, machine learning (ML)-based methods, and deep learning (DL)-based approaches for monitoring, detecting, and classifying sound signals, as well as automated object recognition, have been expected and implemented. The features of acoustic signal are considered for extraction by classical signal processing methods including Short Time Fourier Transform (STFT), Hilbert-Huang Transformation, Wavelet Transform, Limiting Cycle, etc. considered for extracting the features of an acoustic signal. Furthermore, various time or frequency-domain function extraction techniques, such as intrinsic time-scale decomposition, resonance-based sparse signal decomposition, and so on, are used. Nonetheless, they did not thoroughly understand the structure features, which resulted in significant issues such as poor robustness and a low recognition rate. In feature selection the classical signal processing methods makes life easier. The required features must be chosen under the guidance of an expert. By listening to the reflected sound of marine vessels, trained individuals may identify and recognize their class. This approach usually necessitates more time and effort, as well as being more costly. To monitor, detect, and classify acoustics signals, the researchers are attempting to replace manual feature extraction or traditional signal processing methods with intelligent systems such as neural networks and ML/DL algorithms. [3]

Machine learning, which is a subfield of artificial intelligence, teaches machines how to manipulate data more efficiently. ML models can be defined in three stages as shown in Figure1. They take some data as input, extract some features or patterns and predict the new patterns. These models are applied in almost every industry including speech recognition, social

networking applications, Stock exchanges, biomedical and so on.

C. *Machine Learning Models for Classification*

As we all know that the machine learning algorithms are primarily subdivided into supervised, unsupervised and reinforcement. Types of supervised learning algorithms include classification and regression. Classification algorithms are used when the outputs are restricted to a limited value and regression algorithms are used when the outputs may have any numerical value within a range. Several classifiers such as Support vector machines (SVM), CNN (Convolutional neural networks), K-nearest neighbor, Naïve Bayes, Decision Trees, Multi-Layer Perceptron (MLP) have been developed to classify the dataset and predict the output of new patterns. In the first part of our paper, we discuss briefly about MLP classifier and in the subsequent sections we try to investigate the built MLP model with a dataset and discuss about the measurements of the model with the necessary plots.

II. MULTI LAYER PERCEPTRON

The fundamental component of a neural network is a network with only one node known as the neuron. A neuron contains inputs x_1, x_2, …, x_n. and their corresponding weights w_1, w_2, …, w_n. These inputs are fetched into an activation function, which multiplies the weights and y outputs by an activation function.

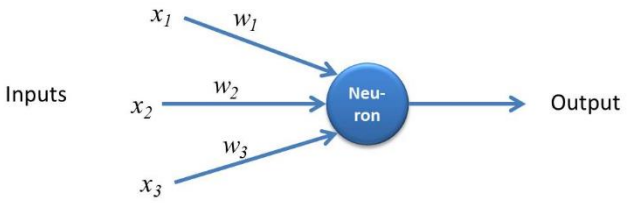

Figure 2 Single Layer Perceptron

A multilayered perceptron consists of at least 3 nodes, a form of neural network. Furthermore, with the exception of the input node, each node of the multilayer sensor is a neuron that uses a nonlinear activation mechanism. The input layer, hidden layer and output layer are three layers of node.

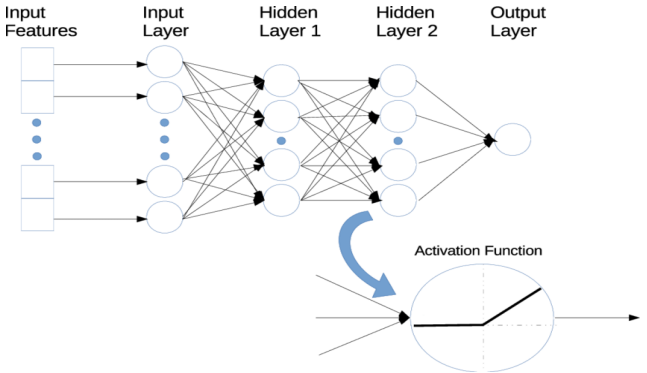

Figure 3 Multi-Layer Perceptron

The input layer, two hidden layers and one output layer of a Multilayer Perceptron are shown in figure 3. The weighted inputs are added together and then passed through an activation function, also known as a transfer function.

Activation Function:

An activation function is a direct mapping of summed weighted input to neuron output. It is referred to as an activation function since it defines the activation threshold of the neuron as well as the intensity of the output signal.

Previously, simple linear functions were used, in which the neuron would output a value of 1.0 if the summed input was higher than a threshold such as 0.5, otherwise a value of 0.0. Non-linear activation functions have traditionally been used. This enables the network to integrate the inputs in more complex ways, resulting in a more robust capability in the functions it can model. Few non-linear activation functions are discussed below.

Sigmoid or Logistic Activation Function:

The logistic function, also known as the sigmoid activation function, has long been a significant activation function for neural networks. The function's input is converted to a value between 0.0 and 1.0. Inputs that are significantly greater than 1.0 are transformed to 1.0, and inputs that are significantly smaller than 0.0 are snapped to 0.0. The function's form for all possible inputs is an S-shape ranging from zero to 1.0 as shown in Figure 4. It was the standard activation on neural networks for a long time, until the early 1990s. The sigmoid function f(x) is given by:

$$f(x)=\text{sigmoid}(x)= 1/ (1+ e^{-x}) \quad (1)$$

Tanh Activation Function:

The hyperbolic tangent function, or tanh for short, is a nonlinear activation function with a similar form that outputs values ranging from -1.0 to 1.0. The tanh function was favored over the sigmoid activation function in the late 1990s and early 2000s because it was easier to train and had better predictive results. [4]. The tanh function g(x) is given by:

$$g(x)= (2/ (1+ e^{-2x}))-1 \quad (2)$$

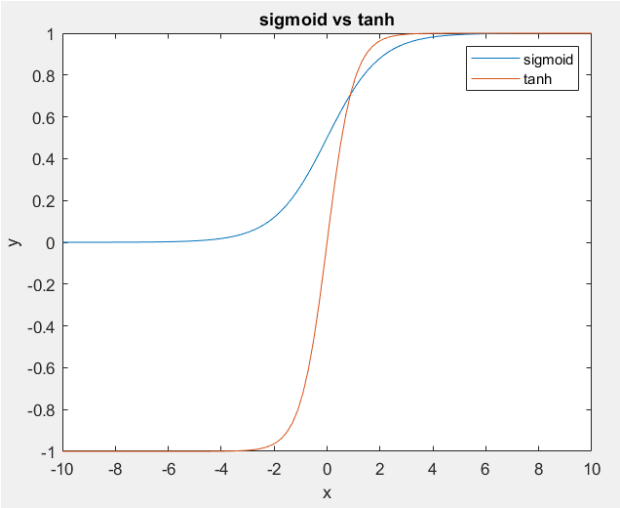

Figure 4 Sigmoid vs Tanh

Saturation is a significant issue with both the sigmoid and tanh functions. This means that for tanh and sigmoid, large values snap to 1.0 and small values snap to -1 or 0.

Furthermore, the functions are only responsive to changes in their input around the mid-point, such as 0.5 for sigmoid and 0.0 for tanh.
The function's minimal sensitivity and saturation occur whether or not the summed activation from the node given as input includes useful information. When the learning algorithm becomes saturated, it becomes difficult for it to continue to adjust the weights in order to improve the model's accuracy. Finally, as hardware capacity increased, very deep neural networks using sigmoid and tanh activation functions could not be easily trained on GPUs. These nonlinear activation functions fail to obtain valuable gradient information in layers deep in large networks. Error is back propagated through the network and used to update the weights. With the derivation of the chosen activation function, the amount of error decreases considerably with each additional layer through which it is propagated. This is known as the vanishing gradient problem, and it prevents deep (multi-layered) networks from effectively learning.
In order to use stochastic gradient descent with backpropagation of errors to train deep neural networks, an activation function is needed that looks and acts like a linear function, but is, in fact, a nonlinear function allowing complex relationships in the data to be learned. The function must also provide more sensitivity to the activation sum input and avoid easy saturation. The solution is to use the rectified linear activation function. [5]

ReLU Activation Function:
ReLU is most frequently used activation function in almost all convolutional neural networks. The function returns 0 if it receives any negative input, but for any positive value z it returns that value back. So, it can be written as a=max (0, z). Graphical representation is given in Figure 5.

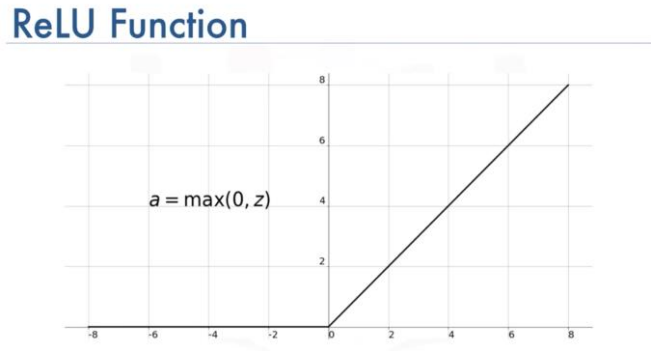

Figure 5 ReLU Function

ReLU is said to have better performance than the logistic function and the hyperbolic tangent function.

Few Other activation functions are given below,

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Figure 6 Other Activation Functions

A neural network executes in two phases: Feed-Forward and Back Propagation.

A. *Feed Forward*

The steps undertaken during the feed-forward process are as follows:

1. The weights are multiplied by the values obtained in the input layer. To prevent null values, a bias is added to the sum of the inputs and weights.
2. Depending on the weights and bias, each neuron in the first hidden layer receives different values from the input layer. The activation function of neurons operates on the value received from the input layer. A step function, sigmoid function, relu function, or tanh function are examples of activation functions.
3. The weights of the second hidden layer are multiplied by the outputs of the first hidden layer neurons, and the results are summed and transferred to the neurons of the subsequent layers. This procedure is repeated until the outer layer has been reached. The actual outputs of the algorithm are calculated at the outer layer.

These three steps make up the feed-forward process. Nevertheless, the expected performance is not always accurate; it may be incorrect, and we must correct it. The goal of an algorithm of learning is to make predictions as accurate as possible. To improve these predicted results, a neural network will then go through a back propagation phase. Back propagation updates the weights of different neurons so that the difference between the desired and predicted output is as small as possible.

B. *Back Propagation*

The following steps are included in the back propagation phase:

1. The difference between the predicted and desired outputs is used to calculate the error. This difference is known as "loss," and the function used to calculate it is known as the "loss function." Loss functions can be of various types, for example, mean squared error or cross entropy functions. Remember that neural networks are supervised learning algorithms that require the desired outputs for a given set of inputs in order to learn from results.

2. If the error has been estimated, the next step is to eliminate it. To do this, the partial derivative of the error function is computed with respect to all weights and biases. This is known as gradient descent. The derivatives can be used to calculate the error function's slope. The derivatives of few activation functions are given in the Figure 6. If the slope is positive, the weight value can be reduced; if the slope is negative, the weight value can be increased. This process is called as weight updation. It can result in fast but also chaotic changes to the network. As a result, the total error is reduced. The learning rate is a configuration parameter that controls how often weights are changed. It is also known as the step size, and it governs the step or shift in network weight for a given error. The optimization function is the function that is used to reduce this error.

This single cycle of feed-forward and back propagation is referred to as an "epoch." This method is repeated until a satisfactory level of accuracy is reached. There is no benchmark for good accuracy; in a perfect world, one would aim for 100 percent accuracy, but this is incredibly impossible to attain for any dataset that isn't trivial.

III. SYSTEM ARCHITECTURE

The system architecture is created to comprehend and emphasize the concepts pursued at the back of classification process in front. This consists of the data collection, merging, data processing, model training, feature extraction after testing, predictions, and results.

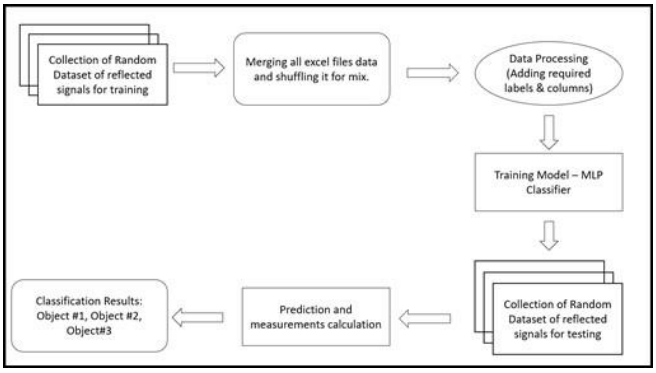

Figure 7 System Architecture

A. *Collecting Data*

We used dataset for reflected waves from object1, object2, object3. This dataset contains 3400 column sound excerpts. There are 3 different classes in the dataset, i.e., Object #1, Object #2, Object #3.

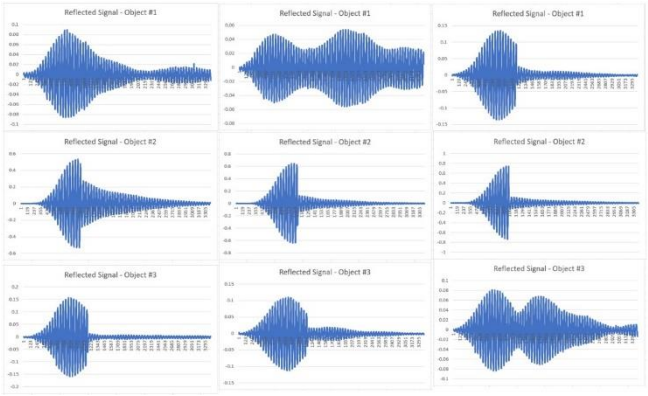

Figure 8 Input Data

The main idea in this paper is to investigate how well sounds can be classified in given different classes. This way, each data row can be classified using classifier model of Multi-Layer Perceptron.

B. *Data Processing*

Data editing is important task when preparing data for its processing in machine learning algorithms. As we all know data gathered in real time scenarios which might consist of noise from external environment which is irrelevant as well as some absurd content values which can hamper the process of getting correct results. [6]

Incomplete data with missing labels which means uneven format also a reason for data processing when we required high quality results.

In this procedure we have done addition of some relevant attributes, labels, and addition of target column for training model which also ensures the error avoidance and productivity increase.

C. *Importing Libraries*

Step one involves the various libraries importing in python environment which will help in numerical and graphical analysis. While working on model we need to plot visuals for different functions which works along with training and testing of algorithms used in this project. There are some of the important libraries such as pandas, numpy, sklearn. Following are patterns to show the libraries imported in this project,

1. Pandas:

Pandas using for data manpulation in rows and columns as we are handling large quantity of data in Excel workbook.

Command: import pandas as pd

2. Numpy:

This is useful for matrix manpulation

Command: import numpy as np

3. SKlearn:

We are using sklearn for MLP classfier. In this, importing components for classifiers and loading matrics for checking accuracy.

Commands:

from sklearn.neural_network import MLPClassifier

```
from sklearn.metrics import classification_report,
confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
```

4. Itertools:

For handing the warnings n project we have added this libraries.

Command: from itertools import cycle

5. Matplot:

Ploting library is useful for visualizing the mathematical solution.

Command: import matplotlib.pyplot as plt

D. Training Model

We are training our model using classifier model of Multi-layer Perceptron in this project. As shown in Figure 9, first layer is taken from data and it is visible layer of model. The second layer of model is called as hidden layer as it is not visible to input, and which can consist multiple layers of neurons in it. The last hidden layer is also known as an output layer which is accountable for generating an output targeted values of given problem.

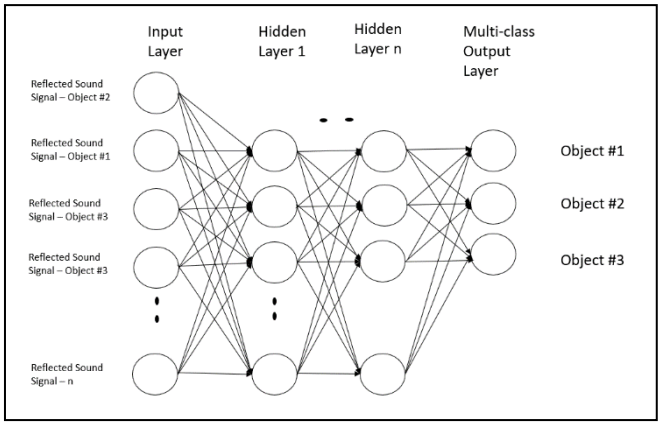

Figure 9 MLP

The actual concept of neural network model which is specifically known as Multi-layer Perceptron classifier, is based on obtaining the linear combinations of specified inputs as resulting features as well as the model target of a nonlinear function of these features. We used the MLPClassifier function in the sklearn Python library for MLP. We selected stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba as the solver forweight optimization (ie, solver = "adam") because of large dataset. We considered hyperparameters, including different hidden layer sizes and activation functions. [7]

E. Prediction

This field consist of replication of biological brain model to resolve the computational problems in machine learning. The purpose of this model creation is not redesigning the real model like brain, but to form an algorithm to resolve the problems with various data structure.

The understanding of neural network comes by their representation ability of learning training data and relate it to the output or target data that we want to predict.

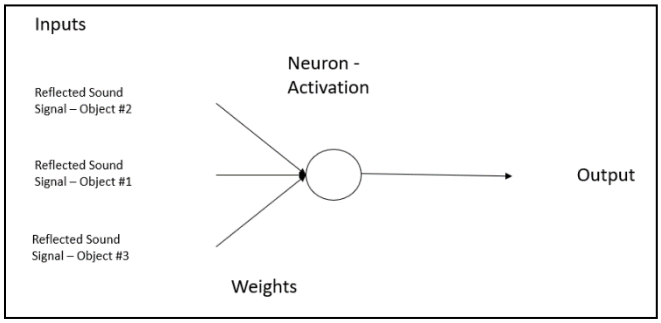

Figure 10 Prediction

As shown in above figure, larger weights prone to complexity of network and fragility of model. It is always expected to keep your model weights minimum to reduce complexity. Activation function which is also called as transfer function in mathematical terms helps all weights to passed through it for further process. In this scenario neural networks trying to learn a mapping. The hierarchical and multilayer structure of MLP classifier helps model's prediction capability.

F. Measurements

Performance of these multilayer neural network classifiers evaluated using parameters such as precision, sensitivity and specificity and the receiver operating curves (ROC). [8]

The evaluation process of classifiers consist number of parameters as follows,

1. true positives (tp)
2. true negatives (tn)
3. false positives (fp)
4. false negatives (fn)

These parameters are used for predicting ability of classifier, i.e., precision, sensitivity, and specificity.

IV. IMPLEMENTATION

This chapter explains the implementation of the Machine Learning project. The project involves creation of a graphical user interface and preprocessing the data which is required by the ML model. The next step involves the design of model using a predefined library and then training and validating the model by splitting the data. Further step is to assess the performance of model by plotting the confusion matrix and ROC curves. The final step involves predicting the output of a new dataset. All the steps are clearly stated in the following segments.

A. GUI Creation

This section gives a brief overview over the architecture and implementation of the Graphical User Interface (GUI).

We designed the GUI as shown in the Figure 11 that allow the user to train and test the ML classifier and to assess the classifier by observing the plots. GUI is built by using the Tkinter framework. It is a cross- platform framework that is

built into the Python standard library. The main window of the GUI is given in the Figure 11.

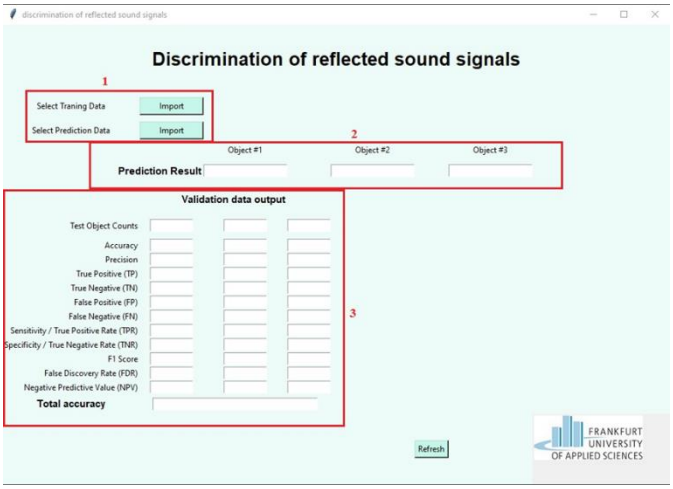

Figure 11 Main Window of GUI

The main window of the GUI is divided into three sections. The functionality of each section is given below.

1. Provides button to select input files for training and doing prediction. On clicking on the "Import" button new window will be displayed with further options.
2. Display's prediction result shows number of objects identified for object types.
3. Display's testing result after completion of training process.

B. Steps for performing training

By clicking on the "import" button for the training data following window will appear to combine multiple data files or to select a single merged file.

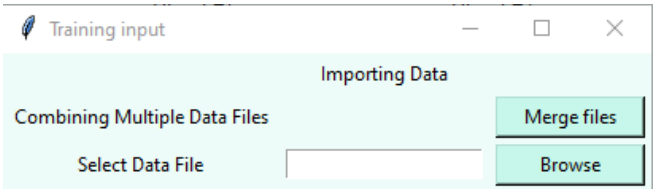

Figure 12 Sub Window 1

On clicking the "Merge files" button a new window will show where option for selection of folder where the files to merge will present. Click "Browse" to select the folder.

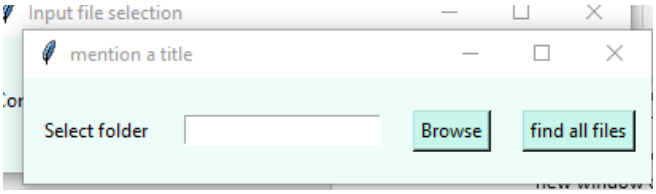

Figure 13 Sub Window 2

Once folder is selected, click on "find all files" this will show all excel files present in the folder with checkboxes and inactive text input boxes. Select the files to merge by clicking on the respective checkboxes, which will allow to enter the object class for each file if it is already known. Then click the "merge files" button to merge files, this will take some time depending on the number of files and their sizes.

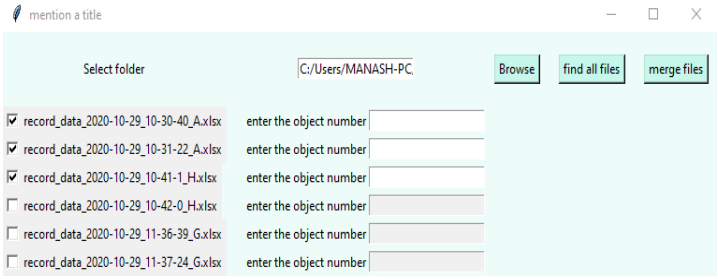

Figure 14 Sub Window 3

Note:

1. The program is build taking in consideration that the object labels are "Object #1", "Object #2" and "Object #3". If any other value is entered the files will be merged with the provided values. Using the merged file for training will also work. But it will generate error for displaying the output on the main window.
2. If files need to be merged without any class label. Leave the object number input boxes empty.
3. If object number is provided, those values will be added at the last column of the output file.
4. Output file will be saved on the selected folder.

"Browse" button opens file explorer window which allows you to select single input excel file for training. After selection of input file for training "save" button will show as below.

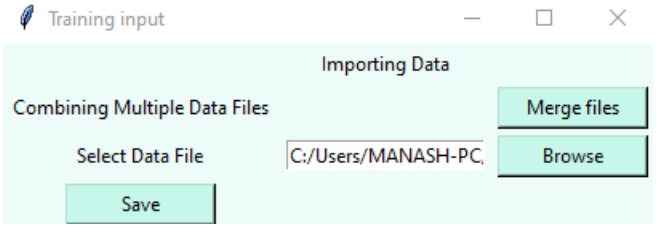

Figure 15 Sub Window 4

Clicking on the "save" button will direct the operation to the main window. Now showing number of rows and columns present on the input file, also providing button "Train" which will initiate the training process.

Note:

1. After clicking "save" button there will be no response for few seconds from the program because the program will be reading the given input excel file and storing that in the Pandas dataframe for further process. This process may take some time.
2. The merged file which will be created for training will contain the class labels on the last column. So, while selecting the number of column range remember to include the last column.

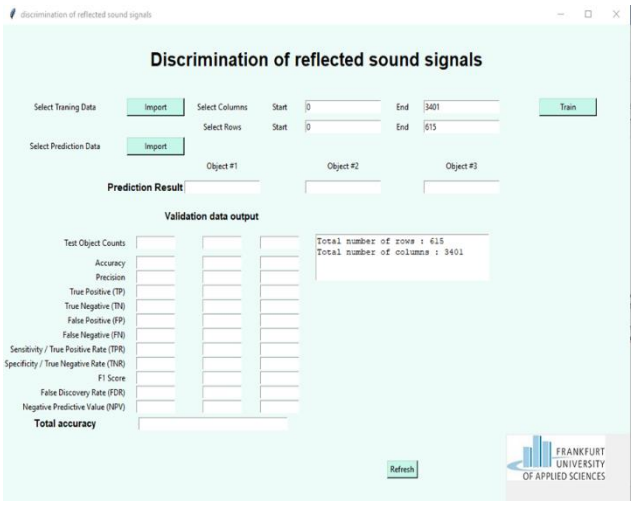

Figure 16 Main Window 1

After clicking on the "Train" button MLP process will be performing the training. After completion of training a popup window will be displayed informing the completion of training. While training, the input dataset will be divided into 80:20 ratio where 80% input will be used for training and 20% will be used for testing. This process will take some time during which program will be unresponsive.

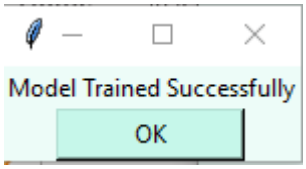

Figure 17 Popup window 1

After clicking "ok" button the results will be displayed on the main window. There it will show how many columns and rows were selected for training and testing. Also, two more buttons will now show "Confusion Matrix" and "ROC Plot" as shown in Figure 18. Trained model output will be saved in a .pkl file. The file will be saved on the same folder from where the input file is selected. Clicking on the "Confusion Matrix" button will display the confusion matrix graph. Clicking on the "ROC Plot" button will show ROC plot.

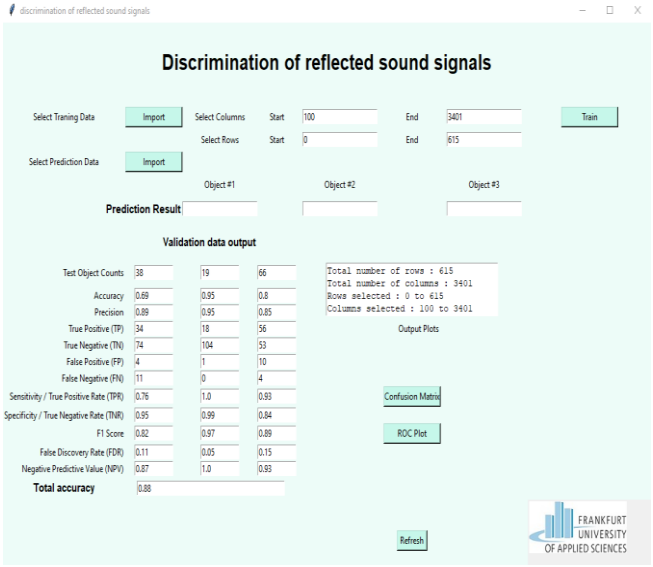

Figure 18 Post Training Window

C. Steps for performing Prediction

Once clicking on the "import" button for the prediction following window will show.

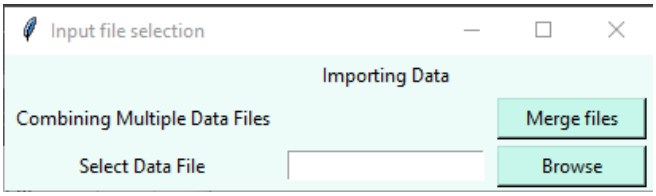

Figure 19 Sub Window 5

Here options to merge multiple data files is provided which can be accessed by clicking on "Merge files" button. This will direct to a new window (follow the same procedure as in training). "Browse" button opens file explorer window which allows you to select single input excel file for training. After selection of the input file for prediction, option to select the trained model file will appear.

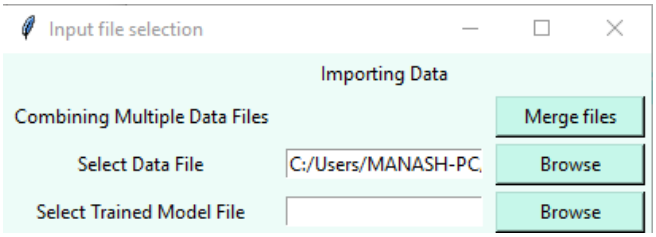

Figure 20 Sub Window 5

After selection of both the files "save" button will appear.

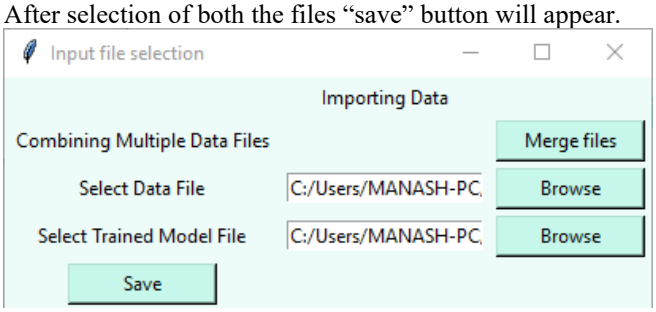

Figure 21 Sub Window 6

After clicking on "save", program will return to main window. Here, it will display number of columns and rows present on the input file. Also, input boxes will appear to take inputs for range of column and row choices. (This will take few seconds as program will be reading the input file.)
After selecting the number of rows and columns to be used. Click on the "Predict" button which will initiate the prediction process. After completion of that a window will be displayed informing of the completion of the prediction process.

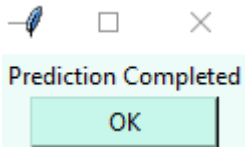

Figure 22 Popup Window 2

After clicking on the “ok” button. Output of the process will be displayed on the main window on “Prediction result” section. New button “view Output” will also show up now.

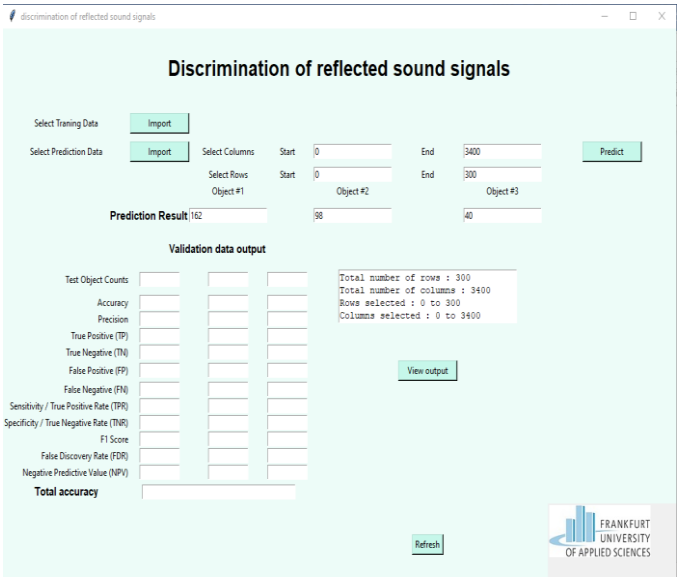

Figure 23 Post Prediction Window

Clicking on the “View output” button will show for which row what is the prediction, and it will also save the output into a new file in the folder from which input file was selected. The process will take few seconds to save the output into a new excel file. The saved file will contain the predicted class on the last column. The final output file is shown in Figure 24.

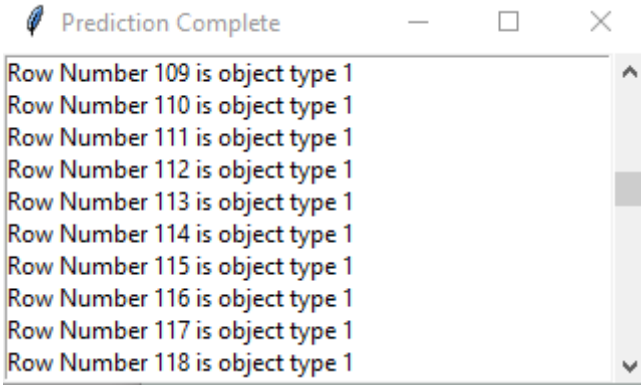

Figure 24 Prediction output

V. RESULTS

A. *Confusion Matrix*

Error Matrix is another name for Confusion Matrix. It is a size of n x n linked with a MLP classifier from sklearn indicates the visualization of learning model with the help of prediction output and original target output of different classes.

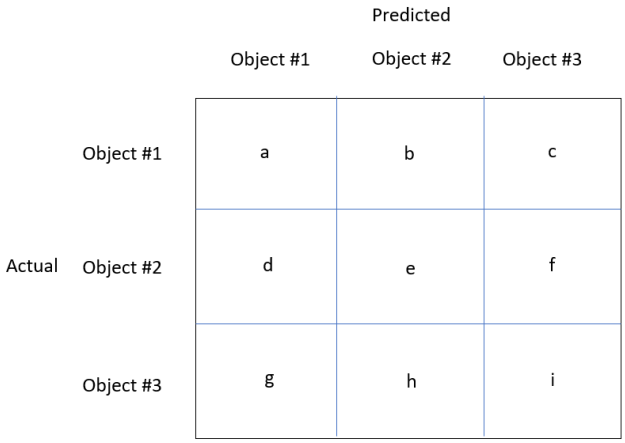

Figure 25 Confusion Matrix

The above figure shows a confusion matrix for n = 3, whose entries have the meaning and which are also useful while calculating other parameters. [9]

B. *True Positive, True Negative, False Positive, False Negative*

The confusion matrix helps us to understand how MLP classifier is performing. This matrix classifing the data into three dfferent objects, i.e. object 1, object 2, object 3 on the basis of multiple input data.

In the evaulation procedure, we are just consdering the results of validation which neither include underlying mechanism of MLP classifier nor the underlying nput training data.

Classification metrics are determined from true positives (TPs), false positives (FPs), false negatives (FNs) and true negatives (TNs), all of which are tabulated in the output plot of confusion matrix as follows,

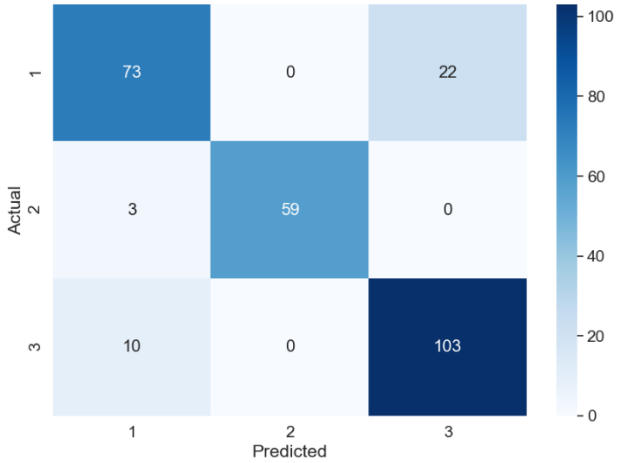

Figure 26 Confusion Matrix Output

The importance of these four parameters will rely on the objective of the classifier and also rely on selection of matric. For object classification test, it determines whether object is correctly predicted of not. [10]

Calculation formulas for TP, TN, FP, FN of multi-class confusion matrix:

Predicted			
C = Actual	C_{11}	. .	C_{pn}
	:	⋱	
	C_{np}		C_{nn}

i = 1, 2 3 (Object 1, Object 2, Object3)

$$TP_i = C_{ii} \quad (3)$$

$$FP_i = \sum_{p=1}^{n} C_{pi} - TP_i \quad (4)$$

$$FN_i = \sum_{p=1}^{n} C_{ip} - TP_i \quad (5)$$

$$TN_i = \sum_{p=1}^{n}\sum_{q=1}^{n} C_{pq} - TP_i - FP_i - FN_i \quad (6)$$

We are able to calculate all these values after converting whole confusion matrix in dataframe.

```
for label in values:
    tps[label] = df.loc[label, label]
    fps[label] = df[label].sum() - tps[label]
    fns[label] = df.loc[label].sum() - tps[label]
for label in set(q):
    tns[label] = len(q) - (tps[label] + fps[label] + fns[label])
```

Figure 27 Code for TP, TN, FP, FN

Here, "df" is dataframe of confusion matrix. These "TP, TN, FP, FN" values we are using for further parameter calculaton.

C. Accuracy, Senstivity, Precsion, F1_score

We showed accuracy, sensitivity, precision and F1 for given classifier. Accuracy is the part of predictions which is having true values. Precision from confusion matrix captures neither TNs nor FNs. [10]

Local Measurements for above parameters are as follows which are different for all three classes [12],

1. $Accuracy_i = \frac{TP_i}{(TP_i + FP_i + FN_i)}$ (7)

Precision is also called as "Positive Predictive Value (PPV)":

2. $Precision_i = \frac{TP_i}{(TP_i + FP_i)}$ (8)

Sensitivity is also called as "Recall", "Hit Rate", "True Positive Rate (TPR)" [12][13]:

3. $Sensitivity_i = \frac{TP_i}{(TP_i + FN_i)}$ (9)

F1 score is nothing but a harmonic mean of precision and sensitivity [13]:

4. $F1\_score_i = \frac{2 * Precision_i * Sensitivity_i}{(Precision_i + Sensitivity_i)}$ (10)

The F1_score is only based on recall and precsion values which means it is not capturing complete confusion matrix and also not cosidering TNs values. [10]

Globally we can find the accuracy of overall confusion matrix with the help of sklearn library.

$$Accuracy = accuracy\_score(y\_test, prediction) \quad (11)$$

Here, "y_test" is actual target output of validaton data and "prediction" is predicted output of validation data.

D. Specificity and Other Parameters

Specificity also called as "Selectivity", "True Negative Rate (TNR)" [12][13]:

1. $\text{Specificity}_i = \frac{TN_i}{(TN_i + FP_i)}$ (12)

Negative Prediction Value (NPV):

2. $NPV_i = \frac{TN}{(TN_i + FN_i)}$ (13)

Miss rate also called as "False Negative Rate (FNR)":

3. $FNR_i = 1 - Sensitivity_i$ (14)

Fall-out also called as "False Positive Rate (FPR)"

4. $FPR_i = 1 - Specificity_i$ (15)

False Discovery Rate (FDR):

5. $FDR_i = 1 - Precision_i$ (16)

False Omission Rate (FOMR)

6. $FOMR_i = 1 - NPV_i$ (17)

E. ROC

Implemented confusion matrix defines the execution of MLP Classifier at one point of operation which is applicable for one time probabilty situaton and position of MLP classifier's weights as well as threads. We can manipulate these weights and thresholds by adding weights into classifier. [11]

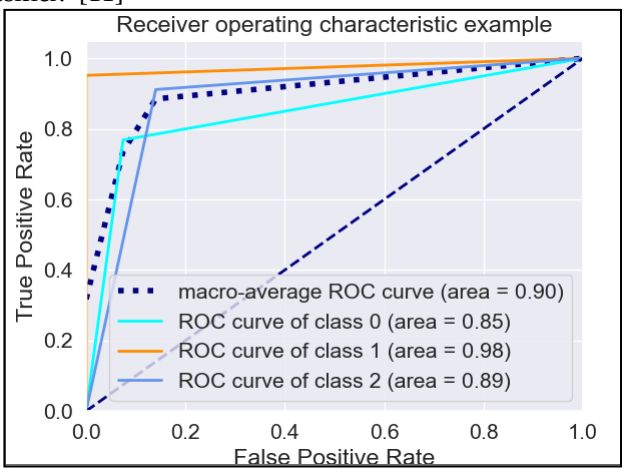

Figure 28 ROC Plot

The classifiers weights as well as the analyzation of respective confusion matrix are benificial for generating independent ROC and group of dimentions [11]. Here in above figure, class 0 is “Object #1”, class 1 is “Object #2”, class 2 is “Object # 3”.

VI. INSTALLATION GUIDE

Below are the following steps to test the project.

1. Install Python from the below link. This project is implemented in Python 3.9. https://www.python.org/downloads/
2. Download and Install any IDE (integrated development environment) such as ‘PyCharm’ or ‘Visual Studio’ Code.
3. Now open the python file(.py) in the IDE.
4. Open the Terminal and import the packages using the following commands. Do it one after another.
    i. pip install pandas
    ii. pip install numpy
    iii. pip install sklearn
    iv. pip install matplotlib
5. After installing the required packages, run the program. Now a GUI window will be opened and follow the instructions which are given in the above implementation section.

VII. CONCLUSION

The goal of this project is the implementation of a Multilayer Perceptron classifier algorithm for discrimination of reflected sound signals in Machine Learning. Almost every machine learning algorithm offers a classification and prediction. Most of them were realized with Python libraries. There are only a few libraries like sklearn, tensorflow, numpy that offers a solution for multilayer perceptron classifier. We are using sklearn algorithm which already supports further processing features, such as model measurements and testing. The goal of this classifier is to discriminate different class of objects with the help of prediction of already learned model with nearby higher precision and accuracy. Based on that, one can say, that the implementation was very successful. The specified parameters of MLP model ensured that the algorithm worked reliably and quickly. In summary, it can be said that the project is carried out successfully according to the specifications.

VIII. ACKNOWLEDGEMENT

We are grateful to **Prof. Andreas Pech** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. We would also like to express our deepest appreciation towards Information Technology Department whose invaluable guidance and constructive criticism supported us in completing this project. At last, we must express our sincere heartfelt gratitude to all the authors who helped us indirectly during this course of work.

IX. APPENDIX

Name	Task
Manash Chakraborty	CODE: • Data processing • Prediction of three classes (Object 1, Object 2, Object 3) REPORT: • Implementation • Installation guide and conclusion
Raghavendra Yarlagadda	CODE: • MLP model creation for train and validation functionality • ROC plots REPORT: • Introduction • MLP (Multilayer perceptron)
Deepali Ashok Betkar	CODE: • GUI layout • Measurement extraction using confusion matrix REPORT: • System Architecture • Results

X. REFERENCES

[1] J. N. M.C. Bisschop, "SOUND REFLECTIONS IN AN AUDIO-BASED GAME FOR VISUALLY IMPAIRED CHILDREN," pp. 4-6, 2015.

[2] J. A. Ratches, "Review of current aided/automatic target acquisition technology for military target acquisition tasks," Optical Engineering , vol. 50, no. 7, 2011.

[3] D. N. a. J. Seok, "A Review on Deep Learning-Based Approaches for," Electronics, 2020.

[4] Y. B. Ian Goodfellow, Deep Learning (Adaptive Computation and Machine Learning series), 2016.

[5] J. Brownlee, "Machine Learning Mistery," 20 08 2020. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/#:~:text=The%20sigmoid%20activation%20function%

[6] Singh, C. (2020). Using Data Analysis and Machine Learning for Studying and Predicting Depression in Users on Social Media (Doctoral dissertation, Carleton University).

[7] Huang, G. H., Lin, C. H., Cai, Y. R., Chen, T. B., Hsu, S. Y., Lu, N. H., ... & Wu, Y. C. (2020). Multiclass machine learning classification of functional brain images for Parkinson's disease stage prediction. Statistical Analysis and Data Mining: The ASA Data Science Journal, 13(5), 508-523.

[8] Singh, R. R., Conjeti, S., & Banerjee, R. (2013). A comparative evaluation of neural network classifiers for stress level analysis of automotive drivers using physiological signals. Biomedical Signal Processing and Control, 8(6), 740-754.

[9] Wakabi-Waiswa, P. P., & Baryamureeba, V. (2011, April). Mining High Quality Association Rules Using Genetic Algorithms. In MAICS (pp. 73-78).

[10] Lever, J., Krzywinski, M., & Altman, N. (2016). Classification evaluation.

[11] Landgrebe, T. C., & Duin, R. P. (2008). Efficient multiclass ROC approximation by decomposition via confusion matrix perturbation analysis. IEEE transactions on pattern analysis and machine intelligence, 30(5), 810-822.

[12] Durairaj, M., & Revathi, V. (2015). Prediction of heart disease using back propagation MLP algorithm. International Journal of Scientific & Technology Research, 4(8), 235-239.

[13] Flach, P. (2019, July). Performance evaluation in machine learning: The good, the bad, the ugly, and the way forward. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 9808-9814).