

Detecting Abnormal Cough and Respiratory Cycles using Machine Learning Techniques

CMPE 684

PROJECT REPORT

UNIVERSITY OF MARYLAND AT BALTIMORE COUNTY

FALL 2022



UMBC

Contributors : Irfan Ali

Instructor : Dr. Mohammad Younis

Table of Contents

ABSTRACT	1
INTRODUCTION AND MOTIVATION.....	2
CHALLENGES	2
LITERATURE REVIEW	3
SENSORS USED	4
ISSUES	5
WearMe Sensor	5
Stemoscope	6
REVIEW OF DATASET	7
CoughVid	7
Coswara.....	8
Sound-Dr	9
No-Co-Co-Da	9
DATASET DEPLOYED	9
Covid-19Cough	10
Virufy.....	11
METHODOLOGY.....	12
DATA EXTRACTION	14
Spectrogram of Cough.....	15
Mel Power Spec	19
FEATURE EXTRACTION	21
MACHINE LEARNING MODELS	24
Gradient Boosting Classifier.....	29
Light Gradient Boosting Machine	31
Extreme Gradient Boosting Machine.....	32
RESULTS	42
CONCLUSION	45
REFERENCES	48

Abstract

Lots of research have put in evidence that the analysis of breath, cough and voice can be used as a precious source of information for pattern recognition algorithms, which are able to extract a great amount of patient data to determinate with high accuracy if the patient suffers from some diseases and disorders. The most part of these diseases are respiratory ones (such as asthma, pneumonia, pertussis, Chronical Obstructive Pulmonary Disease or COPD, tuberculosis and, finally, COVID-19). CNN/ANN have become very popular with the audio classification task due to its impressive performance for speech detection, audio event classification. Scientific evidence shows that acoustic analysis could be an indicator for diagnosing pneumonia or any other respiratory disease.

ANN architecture for audio classification (based on breath and cough) can be engineered for smartphones in a distributed scenario. From analyzing recorded breath sounds on smartphones, it is discovered that patients with respiratory disease have different patterns in both the time domain and frequency domain.

These patterns will be used in this project to diagnose the respiratory cough or respiratory abnormality. Statistics of the sound signals, analysis in the frequency domain will be performed to analyze features. These techniques are assessed and applied and compared for their results. The purpose of this project is to show how machine learning, specifically sound classification, can be used to detect respiratory diseases, mainly COVID-19 from audio recognition of coughs and Heart sounds.

The audio classification can help reduce time and diagnose a diverse range of respiratory conditions, especially now with an ability to provide a potential widespread COVID-19 screening.

INTRODUCTION AND MOTIVATION:

Cough and respiratory sound processing can assist in the early diagnosis of infections such as Covid-19. Even asymptomatic Covid-19 patients can be diagnosed early enough if appropriate speech modeling and signal-processing is applied [1]. In this project report, few machine learning based COVID-19 cough classifiers were compared which can discriminate COVID-19 positive coughs from both COVID-19 negative and healthy coughs recorded on a smartphone via sensors.

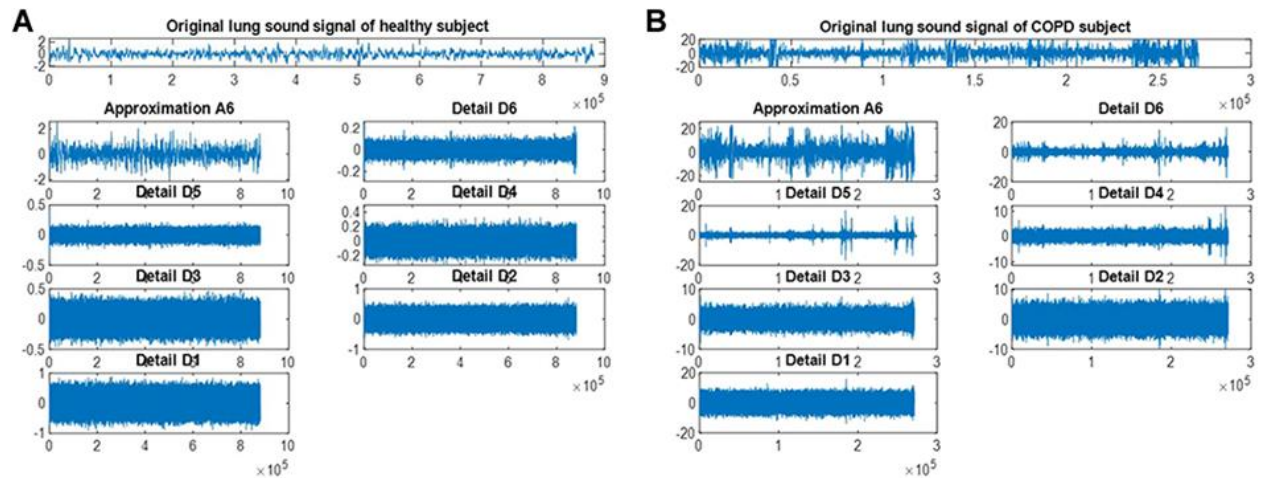


Fig1. Covid/Non Covid Lung Sounds

This type of screening is non-contact, easy to apply, and can reduce the workload in testing centers as well as limit transmission by recommending early self-isolation to those who have a cough suggestive of COVID-19. The datasets used in this study include the publicly available Coswara dataset containing COVID-19 positive and healthy subjects [2]. Moreover some datasets like Coughvid and Virufy were also explored. The datasets indicate that COVID-19 positive coughs are 15%–20% shorter than non-COVID coughs. Dataset skew was addressed by applying the synthetic minority oversampling technique (SMOTE). The machine models implemented on the dataset include, Gradient Boosting Classifier, Light Gradient Boosting Machine, Extreme Gradient Boosting, Random Forest Classifier.

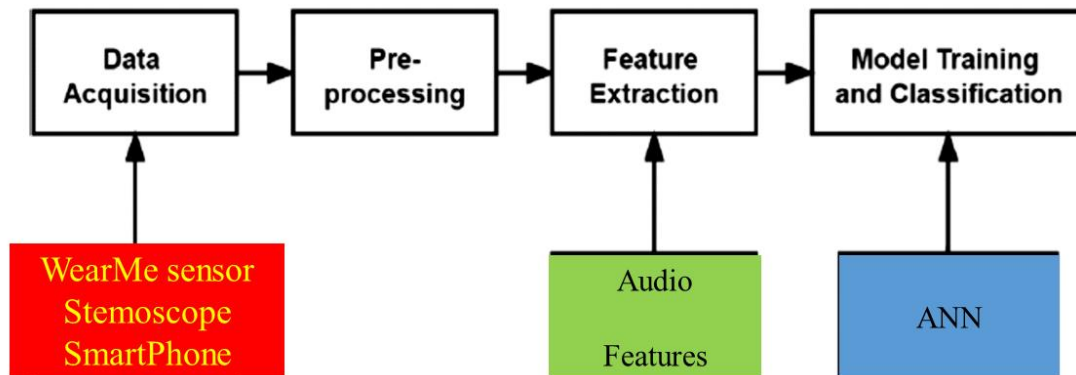


Fig 2. Overall Project Methodology

In this project MFCC (Mel Frequency Cepstral Coefficients) features of audio samples have been used to identify positive and negative cough using various machine learning models. Sound classification – Spectral Analysis, Artificial Neural Network (ANN). Since, the data consists of audio files and csv labels, the project considers building a Convolutional Neural Network (CNN) classification architecture to process them, which considered within Supervised Deep Learning [3].

Challenges

- Differences in quality of audio leads to different inaccurate results
- It was difficult to measure Heart and Lung sounds with the sensors
- The WearMe sensor failed to measure anything, but I could get some response with a Stemoscope
- Potential problems with processing the data (margin of error, equipment malfunction to gain the voice coughs/Heart Sounds)
- Imbalance Dataset

Literature Review

In [1], the authors investigated that patients with COVID-19 have different patterns in recorded breath sounds in both the time domain and frequency domain, which can be used to diagnose the infection of COVID-19. Analysis results show that, an accuracy of over 97% and more than 85% could be achieved with a CNN classifier and kNN with enhanced features respectively [4].

In [2], a unique work on Deep Networks (DN) for COVID-19 is presented, which is inspired by the biological rat's auditory cortex. The researchers have developed Auditory Cortex-Res-Net (AUCCO-Res-Net), deep neural network for COVID-19 sound based recognition from audio tracks of coughs and breaths. This neural network deploys three attention mechanisms called the squeeze and excitation mechanism, the convolutional block attention module, and the sinusoidal learnable attention. The net was tested on the Urban Sound 8K dataset, achieving good accuracy, without any preprocessing of the raw signals [4].

In [3], the authors describe a COVID-19 diagnostic system based on crowdsourced COUGHVID cough database with a sensitivity of 93% and a specificity of 94%. In [4], the authors

discussed the scope of many ML algorithms in COVID-19 prediction like backpropagation, autoencoders (AE), variational autoencoders (VAE), restricted Boltzmann machines, deep belief networks (DBN), convolutional neural networks (CNN), recurrent neural networks (RNN), generative adversarial networks (GAN), caps-nets, transformer, embeddings from language models, bidirectional encoder representations from transformers, and attention in natural language processing.

They also discussed the short-comings of deep learning algorithms such as Auto-ML-Zero, neural architecture search, evolutionary deep learning, their pros, and cons. They talked about RestNet-100 based RCNN, combined with logistic regression (LR), used to identify COVID-19. Auto-ML-Zero can deal with space search bias, and an explanation on shortcomings of evolutionary deep learning algorithms (EAs), such as the stagnation issue and slow convergence to local minima was discussed.

In [5], researchers talk on the largest evaluation dataset currently presented in academic literature. They compare the performance of a self-supervised learning (SSL) and CNN approach to a baseline SVM model. The authors also compare the SSL approach that takes advantage of unlabeled coughs with the more traditional fully supervised CNN approach.

In [6], the authors presented Lung-Pass, an automated lung sound analysis platform consisting of an electronic wireless stethoscope paired with a mobile phone application, which could standardize auscultation process by limiting observer's bias. The Lung-Pass algorithm was developed using neural network technology and trained using sequential derivation sets of lung sound recordings.

Heart rhythm or heart arrhythmias is the phenomenon in which heartbeats don't work properly due to electrical impulses. This causes the heart to irregularly, sometimes too slow, or too fast, depending on the condition. Fluttering and racing heart are the most common arrhythmia symptoms, which is most of the time harmless. Sometimes heart arrhythmias can even be life threatening and may manifest several alarming signs and symptoms. In [7], the authors describe acquisition and analysis of heart activity using AD8232 module, which captures the heart's electrical activity, using the principles of Electrocardiography (ECG).

For the acoustic activity of the heart, a stethoscope and an electret microphone were used to convert the acoustic energy to electrical energy. The signal from each practice is passed through an ADC to translate the signal to a digital signal to allow further operation. Upon acquiring the data, it is then analyzed whether the subject has arrhythmia, murmur, or is normal using a Deep Learning algorithm. The algorithm discussed in this paper uses a Convolutional Neural Network (ConvNet/CNN). The tuned CNN model described in this paper correctly classified human subjects based on 4 classes- normal, abnormal, others, and noisy.

The classification accuracy showed that 80% of the 20 subjects were correctly classified based on their current medical condition. The performance sensitivity of the study was 100%, while performance specificity is 77.78%. The detection of error rate was at 20%.

In [8], the authors present a database of respiratory sounds, namely, cough, breath, and voice and a sound-based diagnostic tool for COVID-19, called Coswara. The sound samples are collected via worldwide crowdsourcing using a website application. The curated dataset is released as open access. The tool is based on prior studies that have shown good accuracy for detecting other respiratory disorders like asthma, pertussis, tuberculosis, and pneumonia.

In [9], the authors analyze AI-guided tools for cough sound screening based on machine learning algorithms which can detect whether a specific cough is ‘dry’ or ‘wet,’ which can help establish the presence of sputum. The authors also talk about RNN to produce specialized sub-models for the SARS CoV-2 classification.

In [10], an automatic non-contact cough detector is presented that can distinguish audio recordings of coughs from snores and other sounds. The researchers implemented two different classifiers: a Gaussian Mixture Model (GMM) and a Deep Neural Network (DNN).

The detected coughs were analyzed and compared in different sleep stages and in terms of severity of Obstructive Sleep Apnea (OSA), along with age, Body Mass Index (BMI), and gender. The database was composed of nocturnal audio signals from 89 subjects recorded during a polysomnography study. The DNN-based system outperformed the GMM-based system, at 99.8% accuracy, with a sensitivity and specificity of 86.1% and 99.9%, respectively.

Sensors Used in the Project



Fig 3. Stemoscope and Lasarrus WearMe sensors

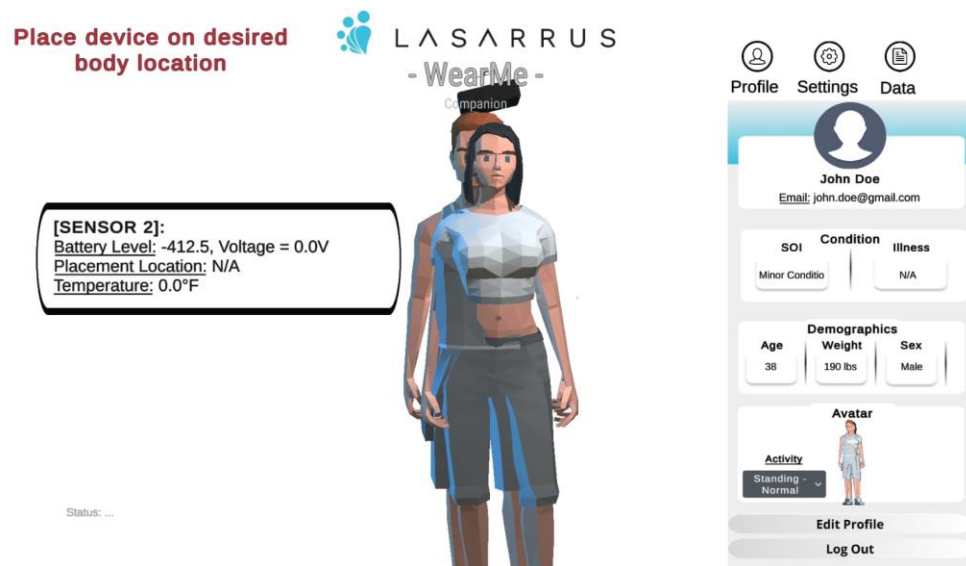


Fig 4. Lasarrus WearMe Sensor Interface

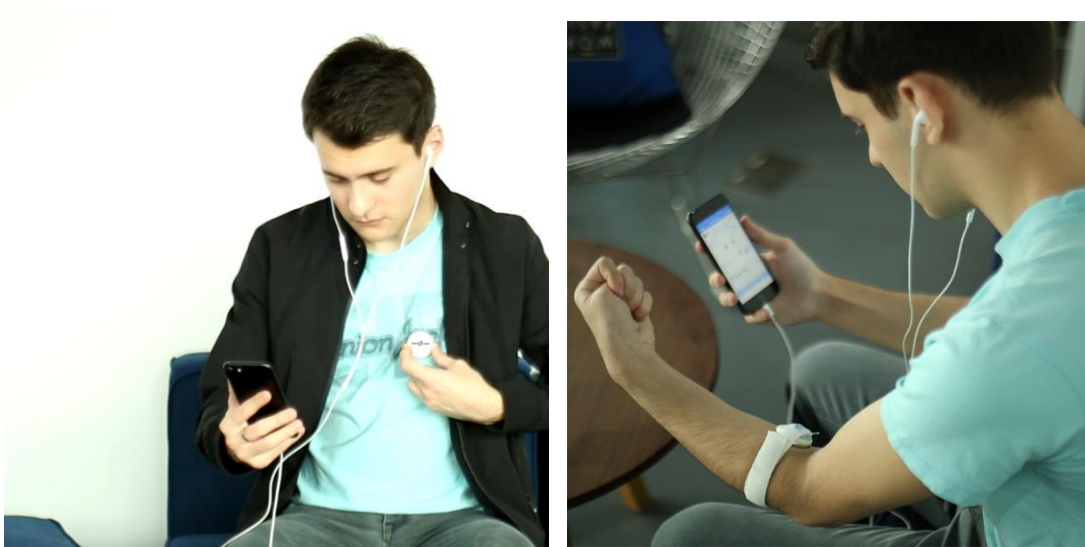


Fig 5. Stemoscope Sensor Interface

- ❖ Stemoscope is the first generation Stemoscope device designed as a non-medical purpose smart listening device [3].
- ❖ The hardware is redesigned to provide better audio performance than the first generation.
- ❖ Stemoscope can detect the sounds transmit the sounds to a smartphone or tablet running the Stemoscope app for further process. The Sounds can be heard by earphones connected to the smartphone or tablet.

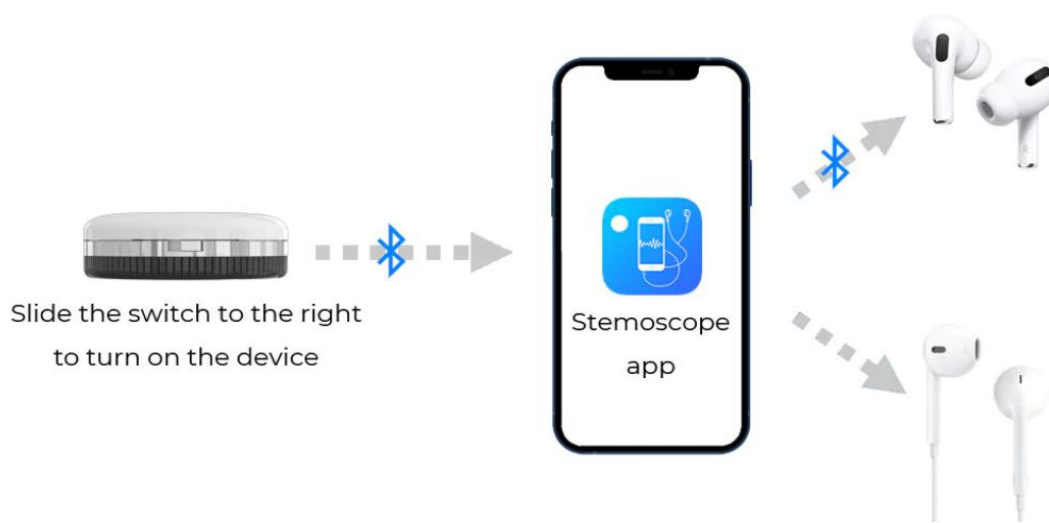


Fig 6. Stemoscope connecting to the smartphone.

SPECIFICATIONS	
Dimension:	38mm (Diameter) x 12mm (Height)
Material:	Chrome plated Zinc alloy body, White plastic cover
Digital Amplification:	YES
App support:	DrStemo app
Frequency range:	20Hz - 1000 Hz
Mode:	Bell, Diaphragm & Extended range
Not a medical device	

Fig 7. Stemoscope Specification

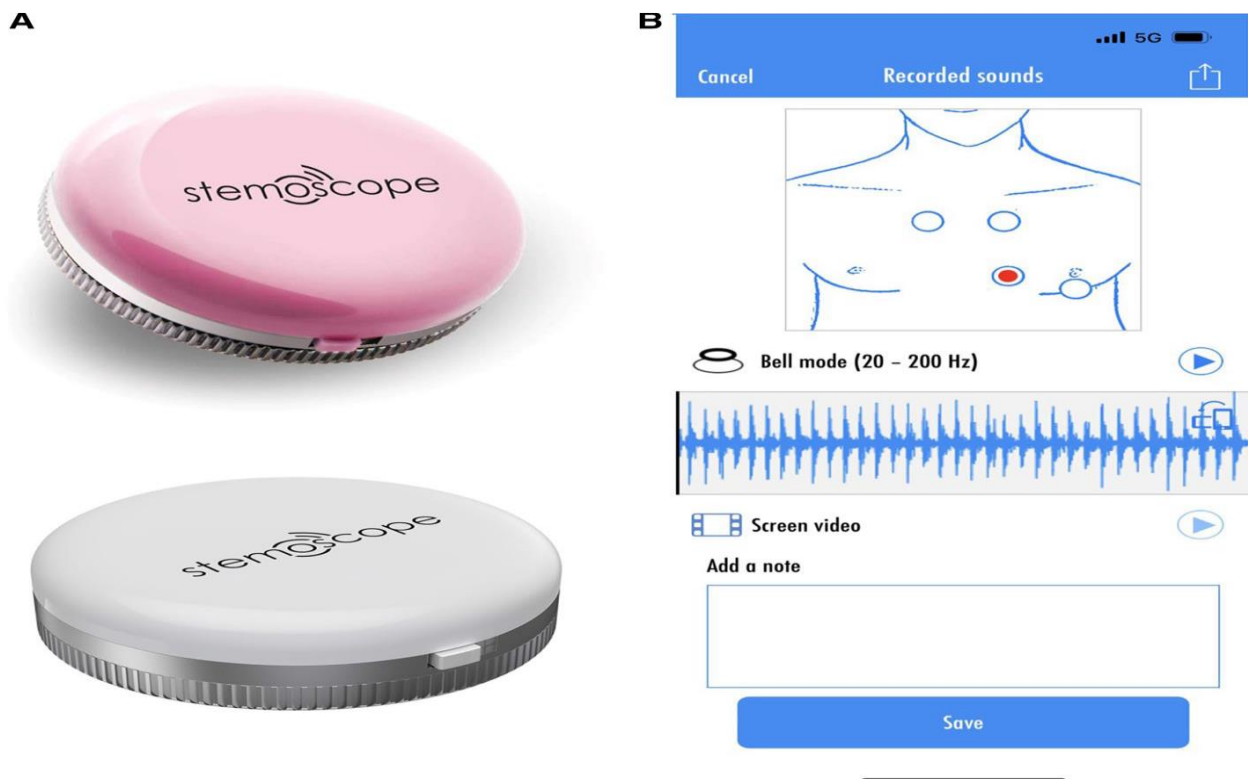


Fig 8. Stemoscope Specification

A Review of Datasets considered for this Project:

1.CoughVid

The Larxel's "Covid-19 Cough Audio Classification" dataset on Kaggle (Larxel, Covid-19 Cough Audio Classification, 2021, Kaggle,

[https://www.kaggle.com/datasets/andrewmvd/covid19-cough-audioclassification?](https://www.kaggle.com/datasets/andrewmvd/covid19-cough-audioclassification?Select=00014dcc-0f06-4c27-8c7b-737b18a2cf4c.webm)

[Select=00014dcc-0f06-4c27-8c7b-737b18a2cf4c.webm](https://www.kaggle.com/datasets/andrewmvd/covid19-cough-audioclassification?Select=00014dcc-0f06-4c27-8c7b-737b18a2cf4c.webm)), with other database sets.

This dataset, also known as CoughVid, is a very popular Dataset for Covid cough sounds. It has over 25,000 crowdsourced cough recordings representing a large range of participants, including ages, genders, geographic locations, and COVID-19 statuses.

A subset of 2,800 recordings were labeled by four experienced physicians to diagnose medical abnormalities present in the coughs. This makes it one of the largest expert-labeled cough datasets used for a plethora of cough audio classification tasks [6].

The audio classification would help reduce time and diagnose a diverse range of respiratory conditions, especially now with ability to provide a potential widespread COVID-19 screening.

2. Coswara

Project Coswara by Indian Institute of Science (IISc) Bangalore is a diagnostic tool for COVID-19 detection using the audio recordings such as breathing, cough and speech sounds of an individual.

Sound samples collected include breathing sounds (fast and slow), cough sounds (deep and shallow), phonation of sustained vowels (/a/ as in made, /i/, /o/), and counting numbers at slow and fast pace. Metadata information collected includes the participant's age, gender, location (country, state/ province), current health status (healthy/ exposed/ positive/recovered) and the presence of comorbidities (pre-existing medical conditions) [7].

Coswara Dataset is publicly open dataset hence was easily available. The preprocessed Coswara dataset, used for feature extraction and classifier training contains total 2278 audio samples out of which 482 are positive. The Subjects in dataset are from five continents: Asia (Bahrain, Bangladesh, China, India, Indonesia, Iran, Japan, Malaysia, Oman, Philippines, Qatar, Saudi Arabia, Singapore, Sri Lanka United Arab Emirates), Australia, Europe (Belgium, Finland, France, Germany, Ireland, Netherlands, Norway, Romania, Spain, Sweden, Switzerland, Ukraine, United Kingdom), North America (Canada, United States), and South America (Argentina, Mexico). Age, gender, geographical location, current health status and pre-existing medical conditions are also recorded. Health status includes 'healthy', 'exposed', 'cured' or 'infected'. Audio recordings were sampled at 44.1 KHz [5].

3. Sound-Dr

This dataset provides an additional dataset on respiration. It is collected with 3,930 sound recordings from 1,300 subjects using a noise reduction method.

4. No-Co-Co-Da

This dataset consists of 73 COVID-19 reflex cough events. The collection of the data is done from identification of interviews with COVID-19 positive individuals to manual cough segmentation [4].

5. EMTprep Lung Sound Database

This YouTube database contains different lung sounds like Bronchial Breath Sounds, Coarse Crackles Lung Sounds, Expiratory Wheezing.

<https://www.youtube.com/watch?v=nLI9sMEPP4w&list=PLXklogn1rJDLeOQ9LvX7MHrj-LNkhHSL0>

6. PhysioNet Dataset

The 2016 PhysioNet/CinC Challenge contains heart sound recordings collected from a variety of clinical or nonclinical (such as in-home visits) environments. These comprise single short recording (10-60s) from a single precordial location [5].

During the cardiac cycle, the heart firstly generates the electrical activity and then the electrical activity causes atrial and ventricular contractions. This in turn forces blood between the chambers of the heart and around the body. The opening and closure of the heart valves is associated with accelerations-decelerations of blood, giving rise to vibrations of the entire cardiac structure (the heart sounds and murmurs). These vibrations are audible at the chest wall and listening for specific heart sounds can give an indication of the health of the heart [6].

<https://physionet.org/content/challenge-2016/1.0.0/>

Datasets Deployed:

COVID-19 Cough Recordings from CovidVid

- The COVID-19 cough audio data is from, the Coswara project combined with data from the Stanford University led Virufy mobile app.
- Our data was split to create a balanced train/test dataset of 28 audio samples; 14 each for the covid and non-covid labels.
- Along with a validation set of 40 audio samples, two of which are covid labels, with the remaining 38 non covid labels.
- The audio data is loaded and processed into MFCCs, using the Librosa Python library
- The MFCC in the form of a $120 \times 431 \times 1$ tensor along with the corresponding label is passed to the model for training.
- MFCC CNN architectures are very effective for classifying audio data.

Virufy COVID-19 Open Cough Dataset

- Virufy is a volunteer-run organization uniting the world to build a global artificial intelligence (AI) database of crowdsourced cough sounds to identify patterns that signify respiratory diseases, such as COVID-19.

- All data is labeled with COVID-19 PCR test status, along with patient demographics as can be found in labels.csv.
- The data is preprocessed and labeled with COVID-19 status (acquired from PCR testing), along with patient demographics (age, gender, medical history).
- It comprises of 121 segmented cough samples from 16 patients.
- The data directory contains two different folders, one with the original cough audio recordings, and the other with the segmented versions of the coughs.
- The segmented coughs were created by identifying periods of relative silence in the recordings and separating coughs based on those silences.

Methodology

Audio Signals that were present in the dataset in the form of .wav files. These audio files could not be directly fed to a model. This left two major approaches – plotting the audio signal or extracting certain characteristic features from the signals. To implement this project, the latter approach was employed.

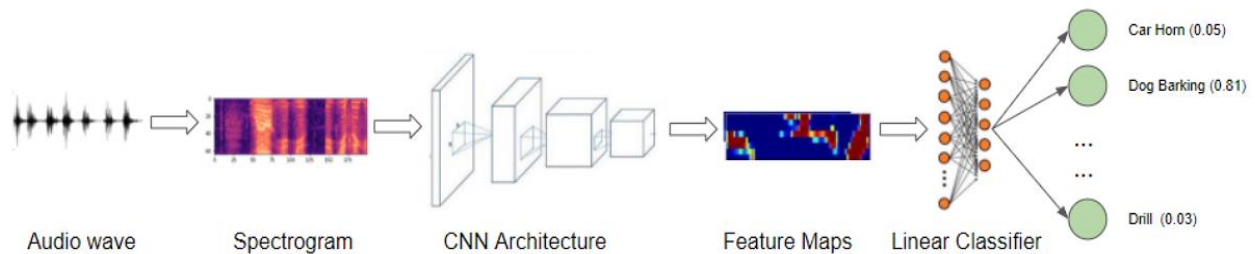


Fig 9. Architecture

COVID-19 diagnosis is done by extracting some features from the audio signals: - Mel-frequency cepstral coefficients (MFCC)(20 in number), Spectral Centroid, Zero-Crossing Rate, Chroma Frequencies, Spectral Roll-off. These extracted features are then used to train an Artificial Neural Network that has a sigmoid activation at the output layer making binary predictions.

Data Extraction and Pre-processing

The raw cough audio recordings from dataset have the sampling rate of 44.1 KHz and is subjected to some simple pre-processing steps, described below. The project aims at finding the mfcc coefficients of each sample. In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency [6].

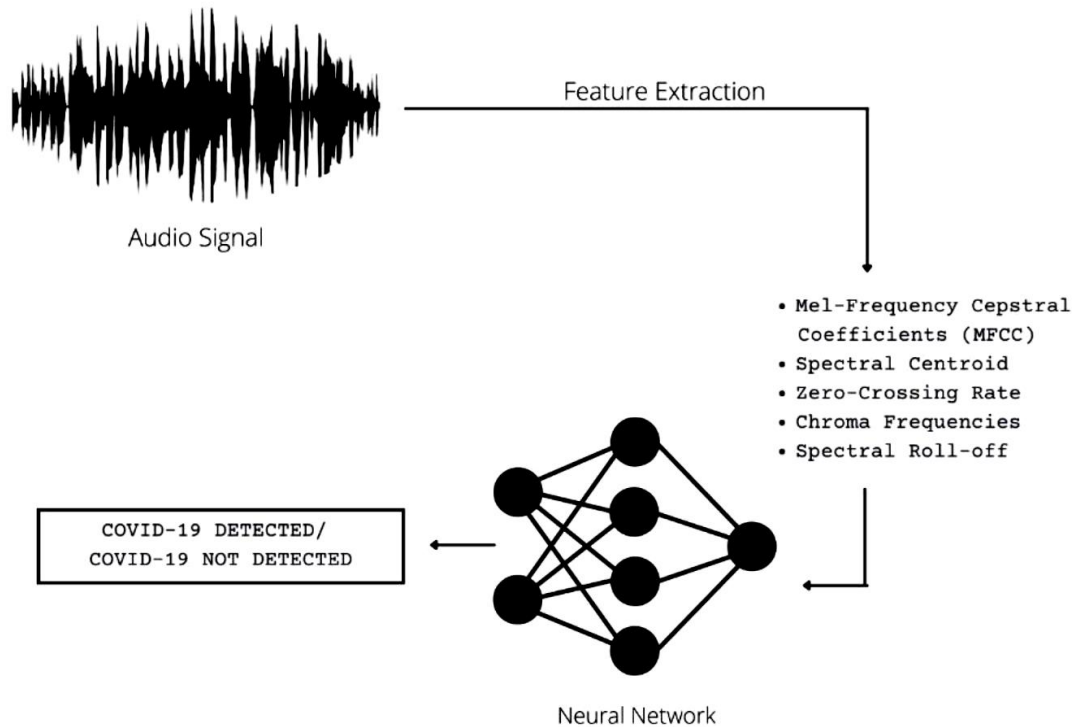


Fig 10. Model Overview.

- Audio Signals dataset are within .wav files or .mp3 files.
- It comprised of cough audio recordings of a pool of 170 COVID-19 positive and negative patients.
- The audio file paths were stored in a .csv file and were tagged as 'covid' and 'not_covid'.
- These audio files are fed to the model after extracting certain characteristic features from the signals.

Spectrogram of Cough Audio Sample

A spectrogram is a visual way of representing the signal strength, or “loudness”, of a signal over time at various frequencies present in a particular waveform. Not only can one see whether there is energy at, but one can also see how energy levels vary over time. A spectrogram is a detailed view of audio, able to represent time, frequency, and amplitude all on one graph. A spectrogram displays changes in the frequencies in a signal over time. Amplitude is then represented on a third dimension with variable brightness or color [7].

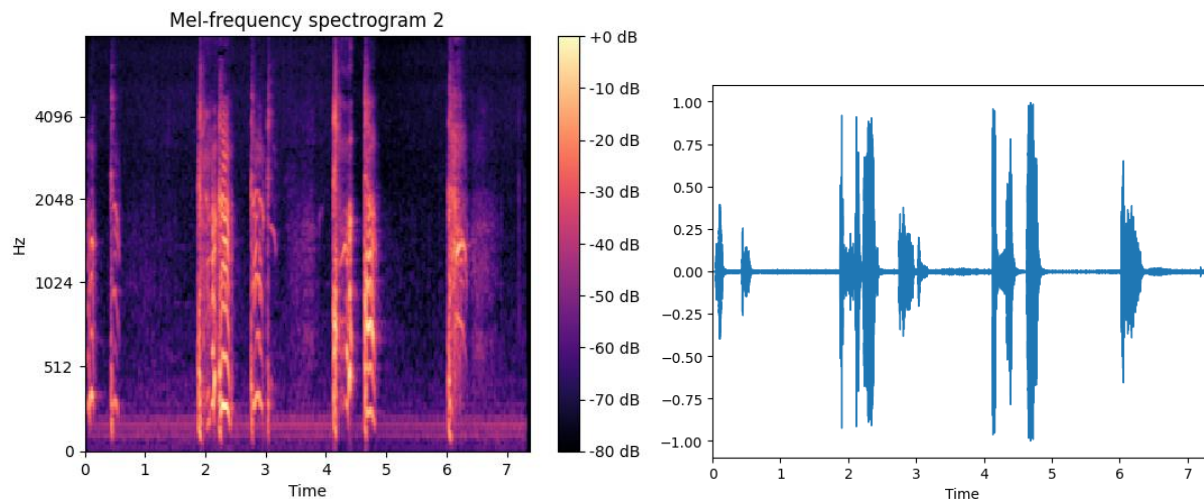


Fig 11. Spectrogram of a Sample

Mel Power Spectrogram

The mel-spectrogram remaps the values in hertz to the mel-scale. The mel-scale is a scale of pitches that human hearing generally perceives to be equidistant from each other. As frequency increases, the interval, in hertz, between mel-scale values (or simply mels) increases. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC.[1]

Mel-frequency cepstral coefficients (MFCCs)

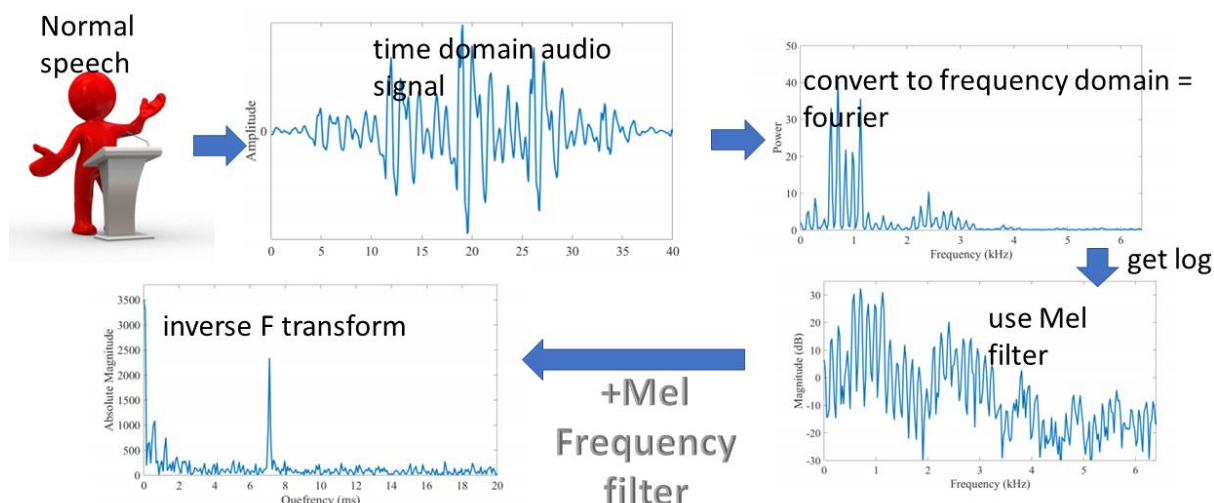


Fig 12. Computing MFCCs

The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly spaced frequency bands used in the normal spectrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

The linear audio spectrogram is ideally suited for applications where all frequencies have equal importance, while mel-spectrograms are better suited for applications that need to model human hearing perception. Studies have shown that humans do not perceive frequencies on a linear scale. We are better at detecting differences in lower frequencies than higher frequencies.

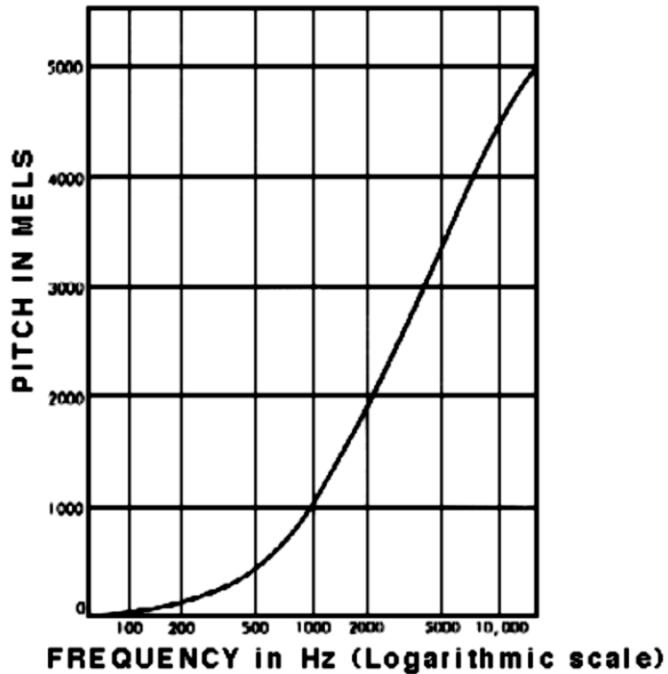


Fig 13. Mel Power Spectrogram of Cough Audio Sample

Feature Extraction

The features extracted from all the audio files of 'heavy cough' include chroma stft, RMS, Spectral centroid, Spectral Bandwidth, Roll off, Zero Crossing Rate, MFCC coefficients.

CHROMA STFT

Chroma STFT The Chroma value of an audio basically represent the intensity of the twelve distinctive pitch classes that are used to study music. They can be employed in the differentiation of the pitch class profiles between audio signals. Chroma STFT is short-term Fourier transformation to compute Chroma features. STFT represents information about the classification of pitch and signal structure.

RMS

Compute root-mean-square (RMS) value for each frame, from a spectrogram. Computing the RMS value from audio samples is faster as it doesn't require a STFT calculation.

Spectral Centroid

The spectral centroid is a measure used in digital signal processing to characterize a spectrum. It indicates where the center of mass of the spectrum is located.

Spectral Bandwidth

The spectral bandwidth or spectral spread is derived from the spectral centroid. It is the spectral range of interest around the centroid, that is, the variance from the spectral centroid.

Roll off

Spectral roll off is the frequency below which a specified percentage of the total spectral energy lies.

Zero Crossing Rate

The zero-crossing rate (ZCR) is the rate at which a signal changes from positive to zero to negative or from negative to zero to positive.

MFCC coefficients

MFCCs have been used very successfully as features in audio analysis and especially in automatic speech recognition. They have also been found to be useful in differentiating dry coughs from wet coughs and classifying tuberculosis coughs. Mel-Frequency Cepstral Coefficients has 39 features. The 39 MFCC features parameters are 12 Cepstrum coefficients plus the energy term. Here we considered 20 MFCC features.

Dataset Balancing

The challenge of working with imbalanced datasets is that most machine learning techniques will ignore, and in turn have poor performance on, the minority class, although typically it is performance on the minority class that is most important. COVID-19 positive subjects are underrepresented in Coswara dataset. To compensate for this imbalance, which can detrimentally affect machine learning, we have applied SMOTE (Synthetic Minority Oversampling Technique) data balancing to create equal number of COVID-19 positive coughs during training. This technique has previously been successfully applied to cough detection and classification based on audio recordings.

Synthetic Minority Oversampling Technique

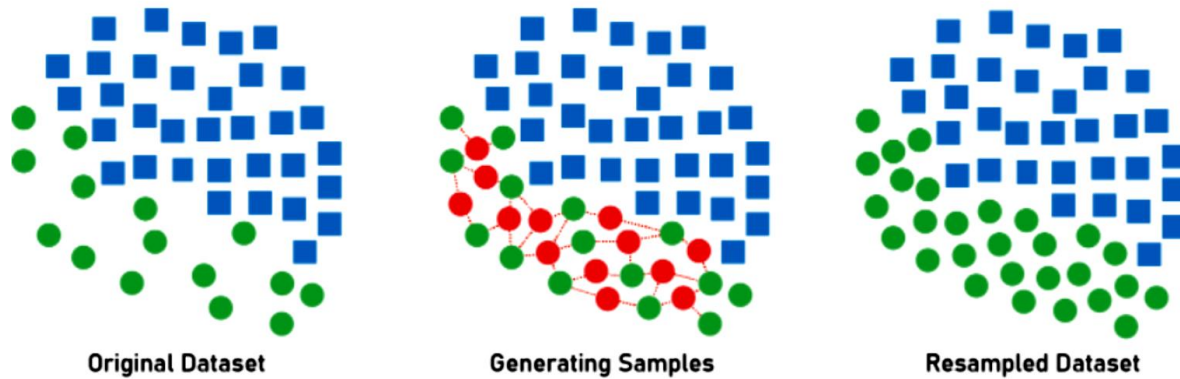


Fig 14. Dataset balancing using SMOTE

SMOTE oversamples the minor class by generating synthetic examples. SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line.

Machine Learning Models

The training data must contain the correct answer, which is known as a target or target attribute. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict), and it outputs an ML model that captures these patterns. Following are the various machine learning models we implemented on the dataset.

Gradient Boosting Classifier

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model. This is done by building a new model on the errors or residuals of the previous model. When the target column is continuous, we use Gradient Boosting Regressor whereas when it is a classification problem, we use Gradient Boosting Classifier [8]. Hence, here classifier is being used. The only difference between the two is the “Loss function”. The objective here is to minimize this loss function by adding weak learners

using gradient descent. Since it is based on loss function hence for regression problems, we'll have different loss functions like Mean squared error (MSE) and for classification, we will have different for e.g log-likelihood [11].

Light Gradient Boosting Machine

Light Gradient Boosted Machine, or LightGBM for short, is an opensource library that provides an efficient and effective implementation of the gradient boosting algorithm, originally developed by Microsoft. LightGBM extends the gradient boosting algorithm by adding a type of automatic feature selection as well as focusing on boosting examples with larger gradients. LightGBM splits the tree leaf-wise as opposed to other boosting algorithms that grow tree level-wise. It chooses the leaf with maximum delta loss to grow [13]. Since the leaf is fixed, the leaf-wise algorithm has lower loss compared to the level-wise algorithm. Leaf-wise tree growth might increase the complexity of the model and may lead to overfitting in small datasets [12].

LightGBM implements a highly optimized histogram-based decision tree learning algorithm, which yields great advantages on both efficiency and memory consumption

Advantages

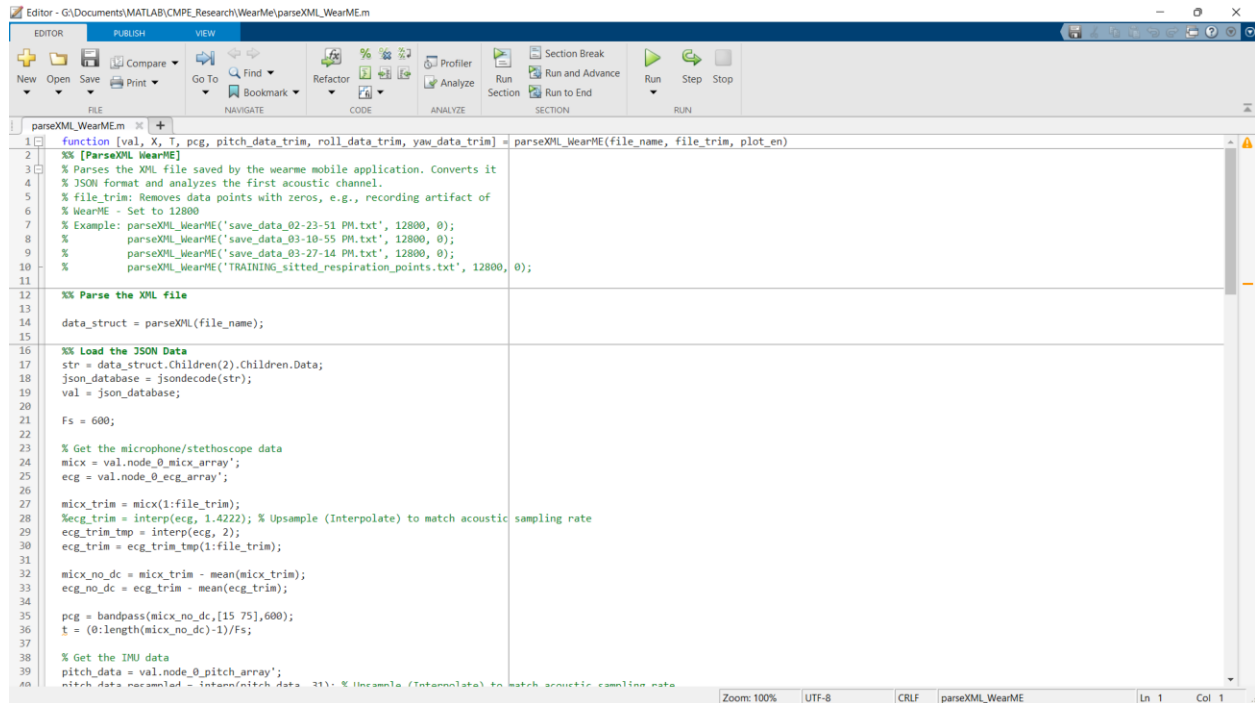
- Faster training speed and higher efficiency
- Lower memory usage
- Better accuracy
- Support of parallel and GPU learning
- Capable of handling large-scale data

Extreme Gradient Boosting

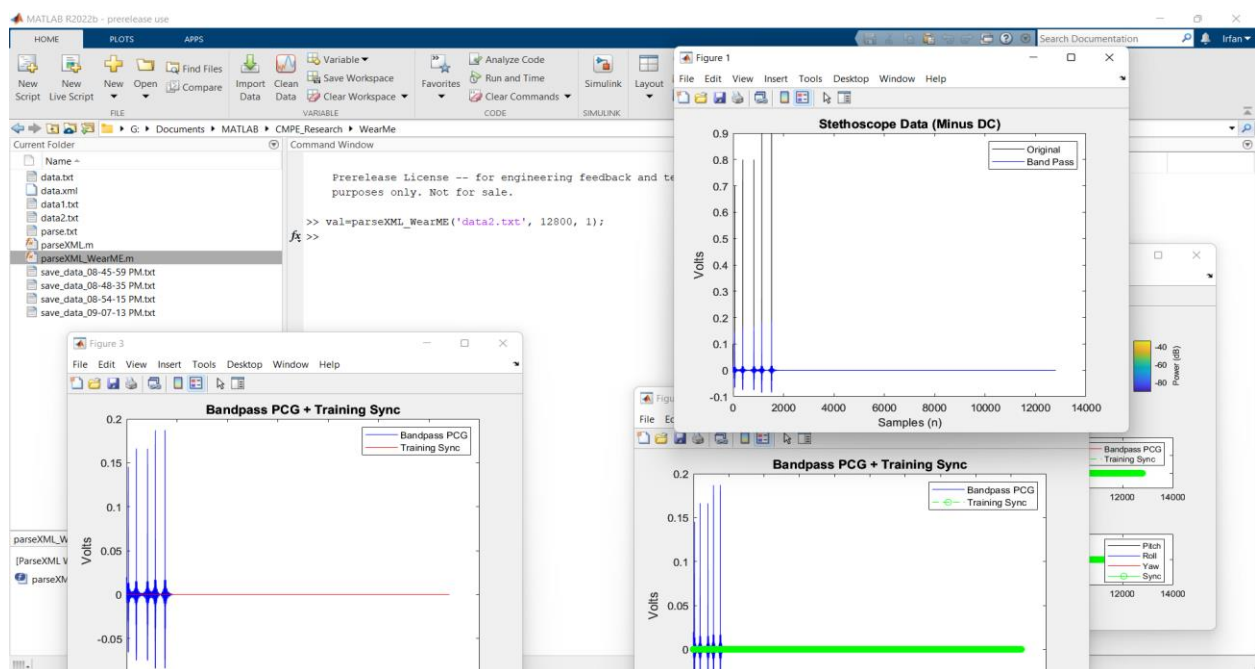
Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models. When using gradient boosting for regression, the weak learners are regression trees, and each regression tree maps an input data point to one of its leafs that contains a continuous score [11]. XGBoost minimizes a regularized (L1 and L2) objective function that combines a convex loss function (based on the difference between the predicted and target outputs) and a penalty term for model complexity (in other words, the regression tree functions). The training proceeds iteratively, adding new trees that

predict the residuals or errors of prior trees that are then combined with previous trees to make the final prediction. It's called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models [13].

Extracting Lung sounds from WearMe Sensor



```
1 function [val, X, T, pcg, pitch_data_trim, roll_data_trim, yaw_data_trim] = parseXML_WearME(file_name, file_trim, plot_en)
2
3 %% [ParseXML_WearME]
4 % Parses the XML file saved by the wearme mobile application. Converts it
5 % JSON format and analyzes the first acoustic channel.
6 % file_trim: Removes data points with zeros, e.g., recording artifact of
7 % WearME - Set to 12800
8 % Example: parseXML_WearME('save_data_02-23-51 PM.txt', 12800, 0);
9 % parseXML_WearME('save_data_03-10-55 PM.txt', 12800, 0);
10 % parseXML_WearME('save_data_03-27-14 PM.txt', 12800, 0);
11 % parseXML_WearME('TRAINING_sitted_respiration_points.txt', 12800, 0);
12
13 %% Parse the XML file
14 data_struct = parseXML(file_name);
15
16 %% Load the JSON Data
17 str = data_struct.Children(2).Children.Data;
18 json_database = jsondecode(str);
19 val = json_database;
20
21 Fs = 600;
22
23 % Get the microphone/stethoscope data
24 micx = val.node_0_micx_array';
25 ecg = val.node_0_ecg_array';
26
27 micx_trim = micx(1:file_trim);
28 %ecg_trim = interp(ecg, 1.4222); % Upsample (Interpolate) to match acoustic sampling rate
29 ecg_trim_tmp = interp(ecg, 2);
30 ecg_trim = ecg_trim_tmp(1:file_trim);
31
32 micx_no_dc = micx_trim - mean(micx_trim);
33 ecg_no_dc = ecg_trim - mean(ecg_trim);
34
35 pcg = bandpass(micx_no_dc, [15 75], 600);
36 t = (0:length(micx_no_dc)-1)/Fs;
37
38 % Get the IMU data
39 pitch_data = val.node_0_pitch_array';
40 pitch_data_parallel = interp(pitch_data, 31); % Upsample (Interpolate) to match acoustic sampling rate
```



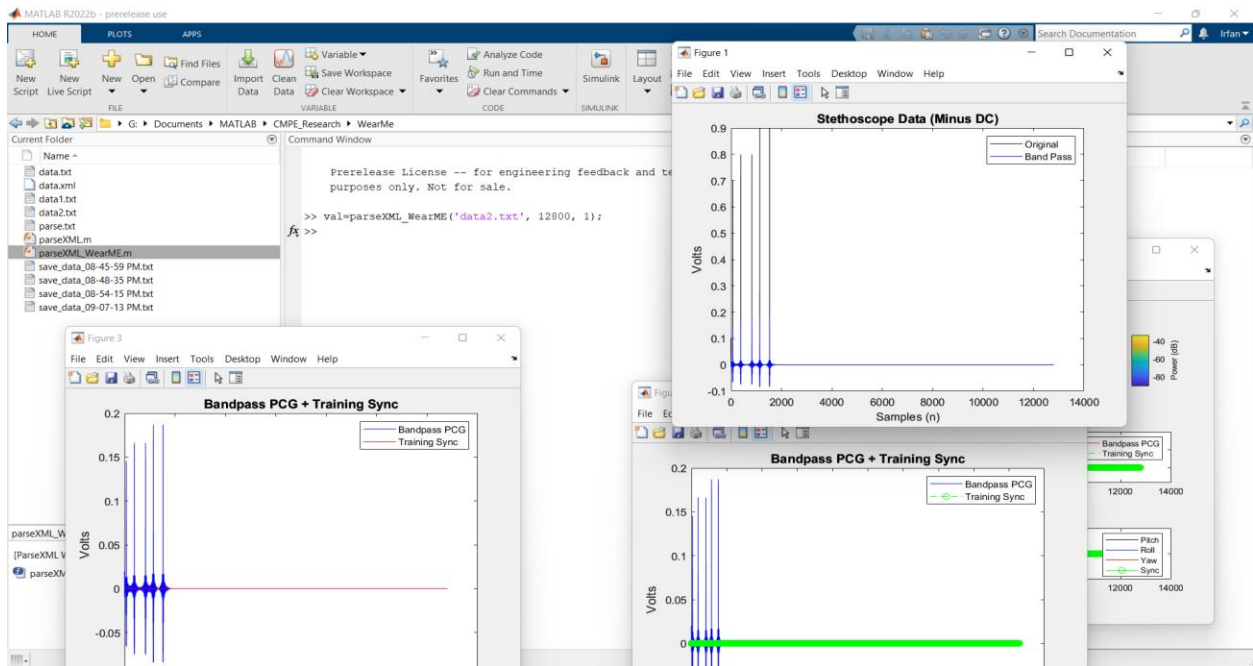


Fig 15. Matlab Interface for WearMe Sensor

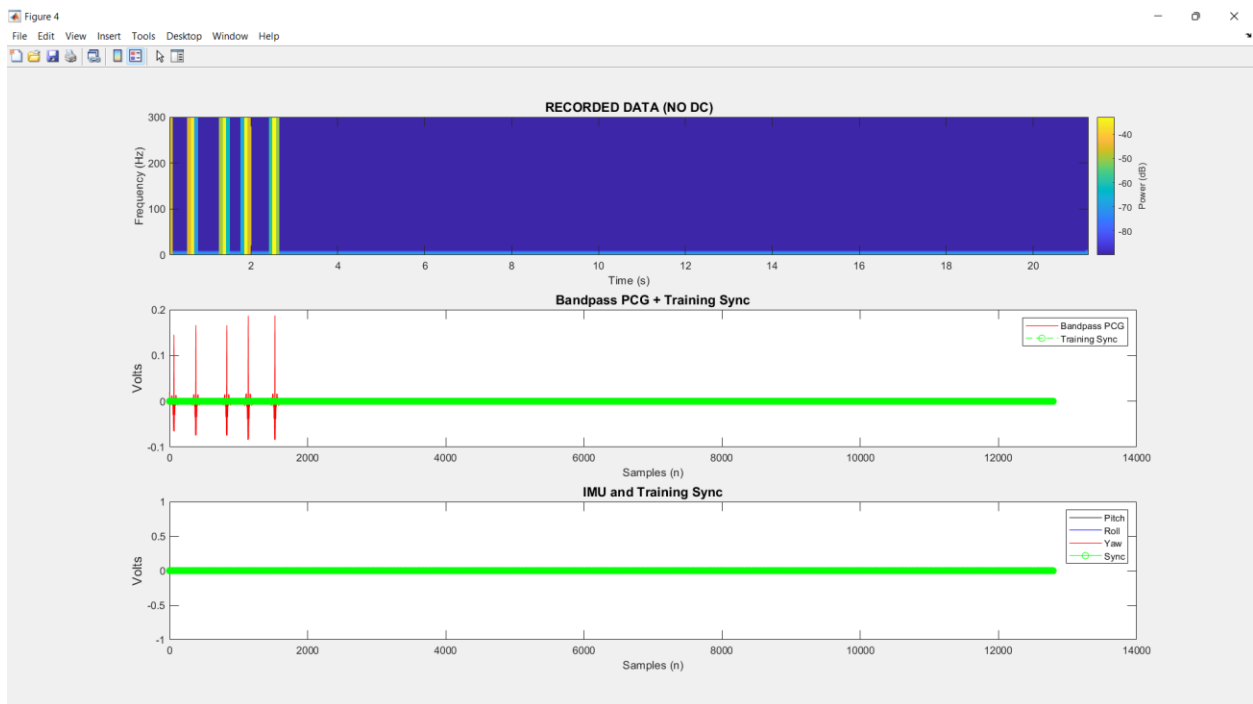


Fig 16. Matlab Interface for WearMe Sensor

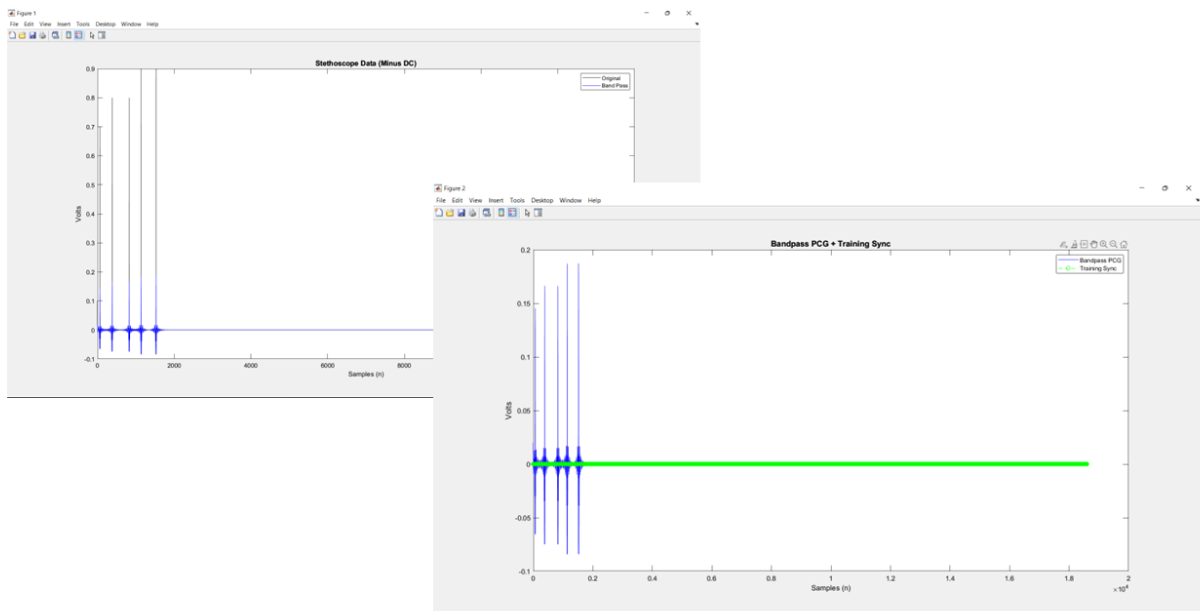


Fig 17. Extracting Audio from WearMe Sensor

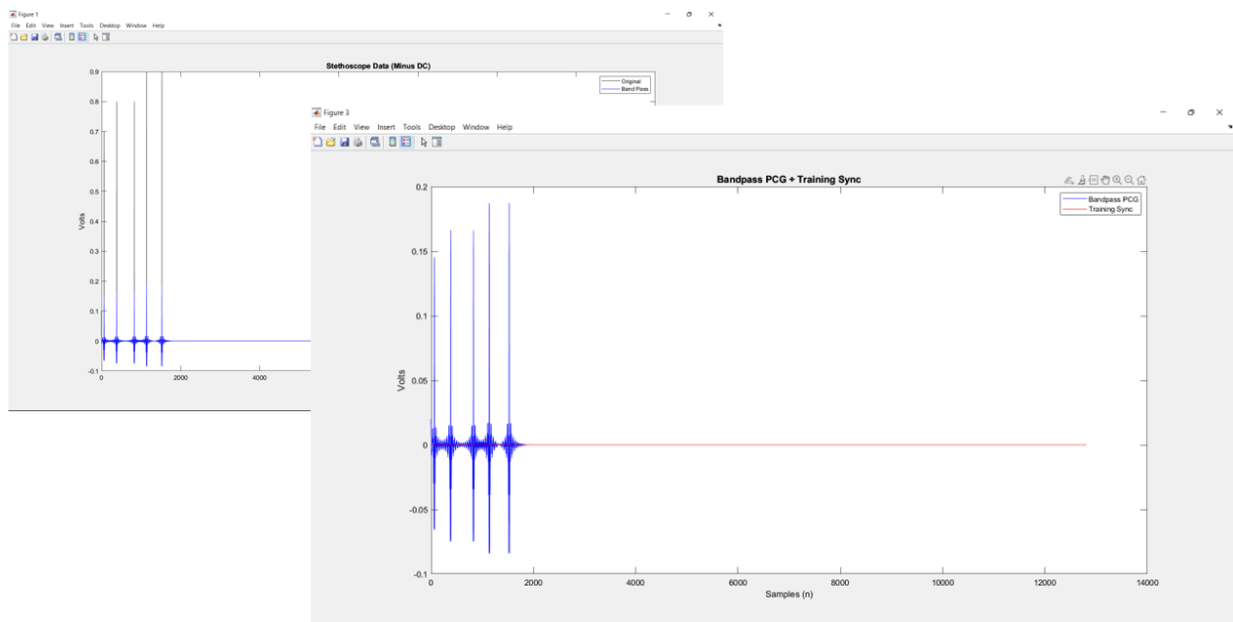


Fig 18. Extracting Audio from WearMe Sensor

Extracting Heart and Lung Sounds from Stemoscope

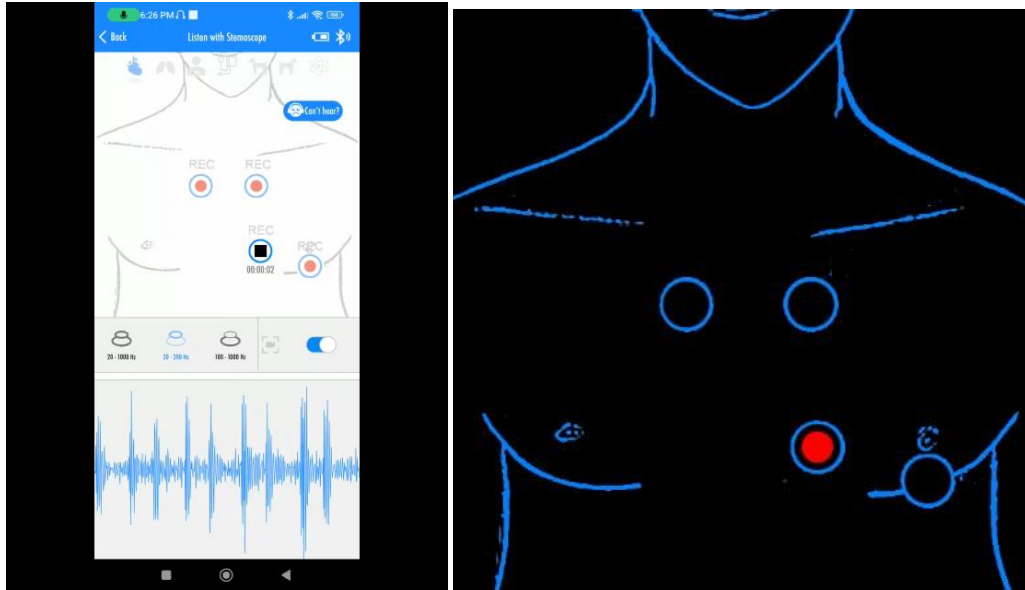


Fig 19. Extracting Audio from Stemoscope

```
In [39]: # Extracting Features from Stemoscope Lung Data
#Loading CSV file
train_csv1 = pd.read_csv("stemoscope.csv") #Read csv.file
dataset1 = "stemoscope.csv" # Path
#anothercsv = pd.read_csv('cough_dataset.csv')
print (train_csv1.head(5)) # Get the first 4 values
#print (anothercsv)

      file_properties      class
0      lungBackR.wav  not_covid
1  lungBackRight.wav  not_covid
2  LungLowerBackR.wav  not_covid
3  LungUpperBackL.wav  not_covid
4  LungUpperBackL.wav  not_covid

In [40]: print (train_csv1['class'].unique()) #Find the class number in the class . column
#print(anothercsv['status'].unique())
print ('\n', train_csv1['class'])

['not_covid']

0    not_covid
1    not_covid
2    not_covid
3    not_covid
4    not_covid
Name: class, dtype: object
```

Extracting Spectrograms of My Lung Sounds

```
In [45]: cmap = plt.get_cmap('inferno')
tot_rows1 = train_csv1.shape[0]
print ('tot_rows',tot_rows1, 'some columns: ', train_csv1.shape[1])
for i in range(tot_rows1): #The range(1) = range(tot_rows)
    source1 = train_csv1['file_properties'][i]
    filename1 = 'Lung_Audio/'+source1
    y,sr = librosa.load(filename1, mono=True, duration=5)
    plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, cmap='magma', sides='default', mode='default', scale='dB');
    #plt.axis('off');
    plt.savefig(f'Spectrogram1/{source1[:-3].replace(".", "")}.png')
    print (source1[:-4]) #Remove the last 4 characters".wav"
    plt.show()
    plt.clf()
```

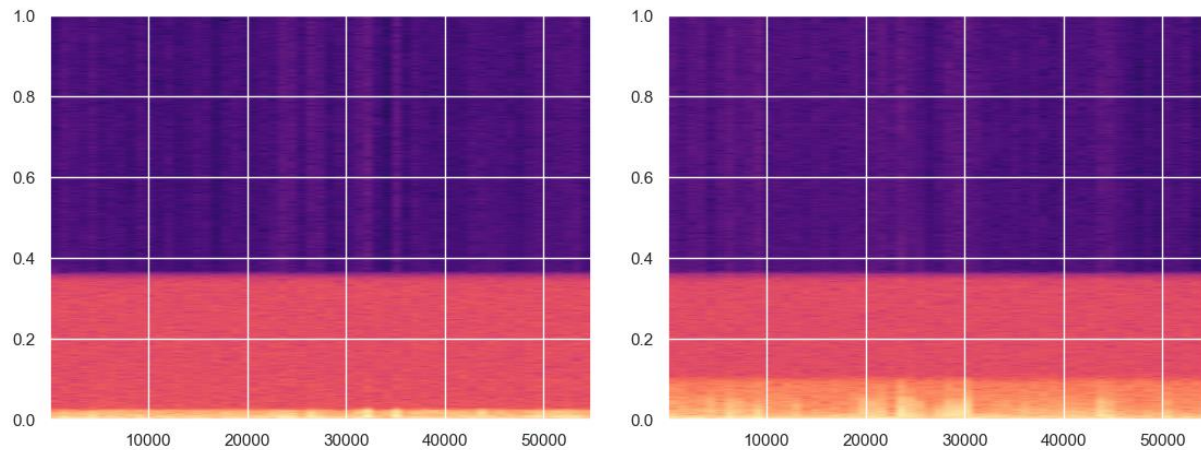


Fig 20. Spectrogram of Lung Audio

```
In [56]: file = open('stemoscope_features.csv', 'w')
with file:
    writer = csv.writer(file)
    writer.writerow(header)
stemoscope_features = pd.read_csv('stemoscope_features.csv')
#print ('data_new_extended\n',data_new_extended)
for i in range(2,4):
    source1 = train_csv1['file_properties'][i]
    print ('source1',source1)
    file_name1 = 'Lung_Audio/'+source1
    label1 = train_csv1['class'][i]
    print ('\nlabel', label1)
    y,sr = librosa.load(file_name1, mono=True, duration=5)
    chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
    rmse = librosa.feature.rms(y=y)
    spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr,hop_length=1024)
    spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr,hop_length=1024)
    rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr,hop_length=1024) #Should have hop-length
    #print ('spec_cent',spec_cent, 'shape:',spec_cent.shape)
    #print ('spec_bw',spec_bw, 'shape:',spec_bw.shape)
    #print ('rolloff',rolloff, 'shape:',rolloff.shape)
    zcr = librosa.feature.zero_crossing_rate(y)
    mfcc = librosa.feature.mfcc(y=y, sr=sr)
    #print ('mfcc',mfcc)
    to_append = f'{np.mean(chroma_stft)} {np.mean(rmse)} {np.mean(spec_cent)} {np.mean(spec_bw)} {np.mean(rolloff)} {np.mean(zcr)}'
    #np.mean average value
    librosa.display.specshow(mfcc, x_axis='time') #Show MFCC
    plt.title('MFCC')
    #plt.show()

    for e in mfcc:
        to_append += f' {np.mean(e)}'
    to_append += f' {label1}'
    value = [str(source1)]
    value.extend(to_append.split())
    file = open('stemoscope_features.csv', 'a')
```

Extracting MFCCs

```
with file:
    writer = csv.writer(file)
    writer.writerow(value)

stemoscope_features = pd.read_csv('stemoscope_features.csv')
print ('stemoscope_features\n',stemoscope_features)
```

```
source1 LungLowerBackR.wav

label not_covid
source1 LungUpperBackL.wav

label not_covid
stemoscope_features
      filename  chroma_stft    rmse  spectral_centroid \
0  LungLowerBackR.wav    0.722649  0.006402      156.865429
1  LungUpperBackL.wav    0.695021  0.008727      130.187844

      spectral_bandwidth  rolloff  zero_crossing_rate    mfcc1    mfcc2 \
0      253.882194    245.438639      0.006508 -695.615356    161.919937
1      231.254804    170.570882      0.005855 -656.856628    136.912598

      mfcc3  ...  mfcc12  mfcc13  mfcc14  mfcc15  mfcc16 \
0  107.377304  ... -1.583107  3.171925  9.414475  11.160958  7.285763
1   94.242516  ...  5.862048  7.953698  12.001055  13.209419  9.674572

      mfcc17  mfcc18  mfcc19  mfcc20  label
0  1.669430 -1.031503  0.326136  3.278772  not_covid
1  3.828084 -0.138912 -0.228061  2.317533  not_covid

[2 rows x 28 columns]
```

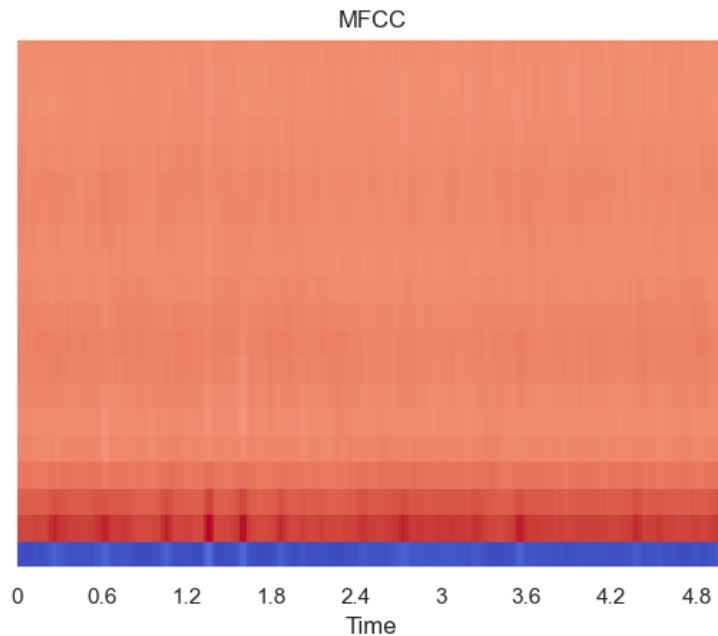


Fig 21. Spectrogram of Lung Audio

Drawing Mel-Spectrograms

```
In [57]: # Drawing mel spectrogram
y,sr = librosa.load('Lung_Audio/LungLowerBackR.wav', mono=True)
fig, ax = plt.subplots(sharex=True, sharey=True)
#draw sound with time axis
librosa.display.waveplot(y, sr=sr, max_points=50000.0, x_axis='time')

S = librosa.feature.melspectrogram(y=y, sr=sr, n_mels=129, fmax=8000)

print ('S.shape',S.shape)
fig, ax = plt.subplots()
S_dB = librosa.power_to_db(S, ref=np.max)
img = librosa.display.specshow(S_dB, x_axis='time',
                                y_axis='mel', sr=sr,
                                fmax=8000, ax=ax)
fig.colorbar(img, ax=ax, format='%+2.0f dB')
ax.set(title='Mel-frequency spectrogram 2')

S.shape (129, 1702)
```

```
Out[57]: [Text(0.5, 1.0, 'Mel-frequency spectrogram 2')]
```

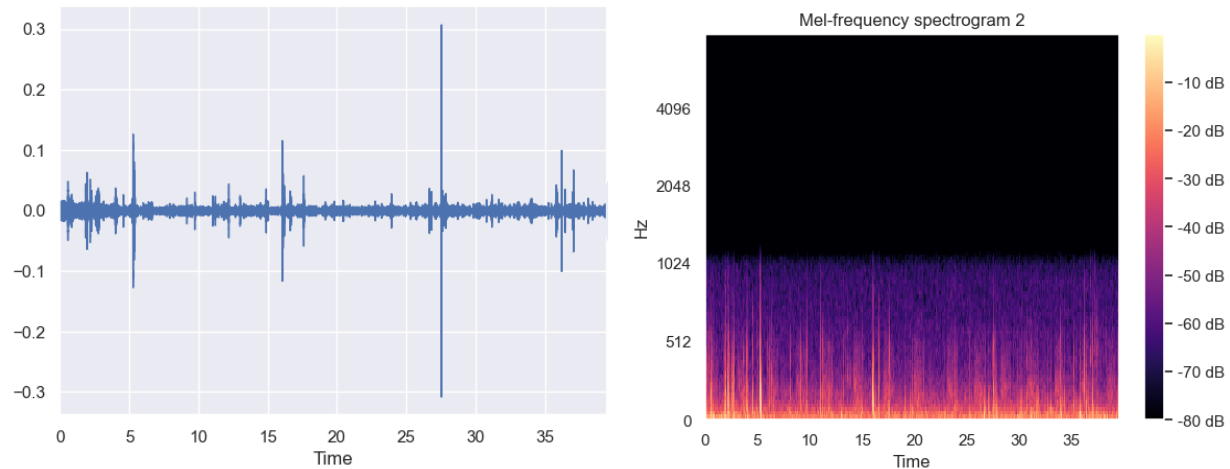


Fig 22. Spectrogram of Lung Audio

Extracting Spectrograms of My Heart Sounds

```
In [58]: # Extracting Features from Stemoscope Heart Data
#Loading CSV file
train_csv2 = pd.read_csv("stemoscope_heart.csv") #Read csv.file
dataset2 = "stemoscope_heart.csv" # Path
#anothercsv = pd.read_csv('cough_dataset.csv')
print (train_csv2.head(7)) # Get the first 4 values
#print (anothercsv)

      file_properties      class
0      LLchest.wav  not_covid
1      LLR.wav      not_covid
2      LowerChestL.wav  not_covid
3      LowerChestLeft.wav  not_covid
4      LowerMiddleChest.wav  not_covid
5      UpperChestRight.wav  not_covid

In [59]: print (train_csv2['class'].unique()) #Find the class number in the class . column
#print(anothercsv['status'].unique())
print ('\n', train_csv2['class'])

['not_covid']

0      not_covid
1      not_covid
2      not_covid
3      not_covid
4      not_covid
5      not_covid
Name: class, dtype: object
```

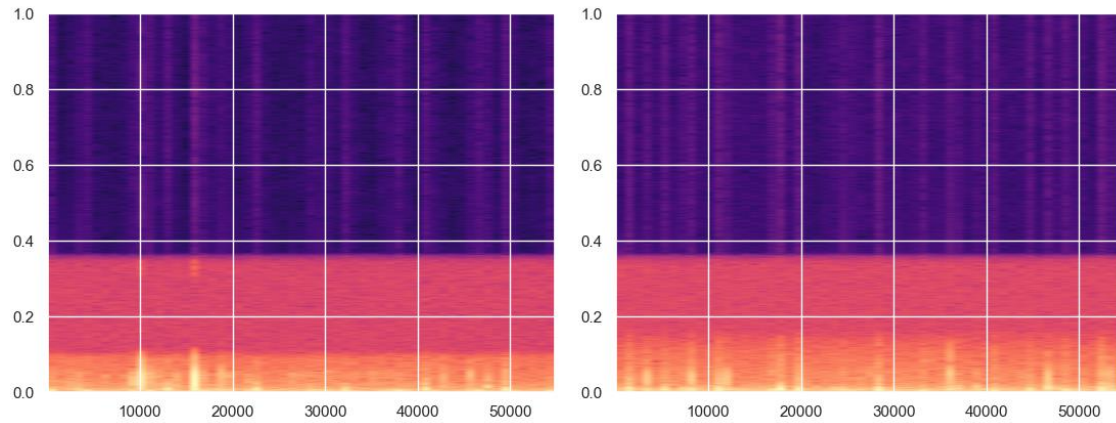


Fig 23. Spectrogram of Heart Audio

```
In [60]: cmap = plt.get_cmap('inferno')
tot_rows2 = train_csv2.shape[0]
print ('tot_rows',tot_rows2, 'some columns: ', train_csv2.shape[1])
for i in range(tot_rows2): #The range(1) = range(tot_rows)
    source2 = train_csv2['file_properties'][i]
    filename2 = 'Heart_Audio/'+source2
    y,sr = librosa.load(filename2, mono=True, duration=5)
    plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, cmap='magma', sides='default', mode='default', scale='dB');
    #plt.axis('off');
    plt.savefig(f'SpectrogramH/{source2[:-3].replace(".", "")}.png')
    print (source2[:-4]) #Remove the last 4 characters".wav"
    plt.show()
    #plt.clf()
```

```
1]: file = open('stemoscope_Hfeatures.csv', 'w')
with file:
    writer = csv.writer(file)
    writer.writerow(header)
    stemoscope_Hfeatures = pd.read_csv('stemoscope_Hfeatures.csv')
    #print ('data_new_extended\n',data_new_extended)
    for i in range(2,4):
        source2 = train_csv2['file_properties'][i]
        print ('source2',source2)
        file_name2 = 'Heart_Audio/'+source2
        label2 = train_csv2['class'][i]
        print ('\nlabel', label2)
        y,sr = librosa.load(file_name2, mono=True, duration=5)
        chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
        rmse = librosa.feature.rms(y=y)
        spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr,hop_length=1024)
        spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr,hop_length=1024)
        rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr,hop_length=1024) #Should have hop-length
        #print ('spec_cent',spec_cent, 'shape:',spec_cent.shape)
        #print ('spec_bw',spec_bw, 'shape:',spec_bw.shape)
        #print ('rolloff',rolloff, 'shape:',rolloff.shape)
        zcr = librosa.feature.zero_crossing_rate(y)
        mfcc = librosa.feature.mfcc(y=y, sr=sr)
        #print ('mfcc',mfcc)
        to_append = f'{np.mean(chroma_stft)} {np.mean(rmse)} {np.mean(spec_cent)} {np.mean(spec_bw)} {np.mean(rolloff)} {np.mean(zcr)}'
        #np.mean average value
        librosa.display.specshow(mfcc, x_axis='time') #Show MFCC
        plt.title('MFCC')
        #plt.show()
```

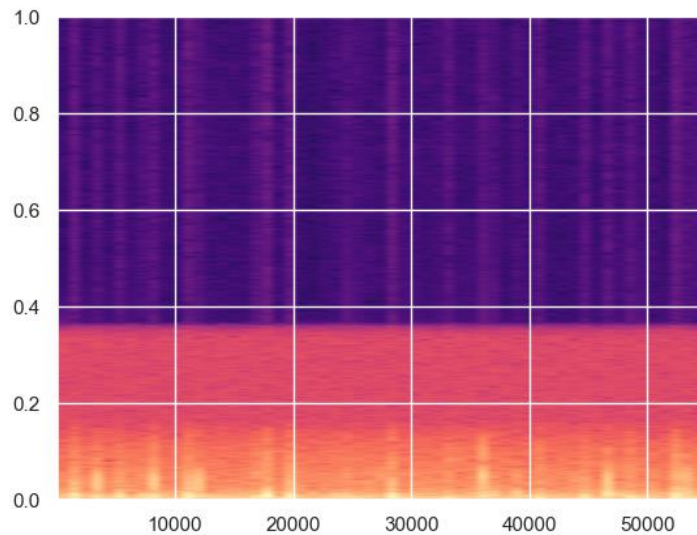


Fig 24. Spectrogram of Heart Audio

```

for e in mfcc:
    to_append += f' {np.mean(e)}'
    to_append += f' {label2}'
    value = [str(source2)]
    value.extend(to_append.split())
    file = open('stemoscope_Hfeatures.csv', 'a')
    with file:
        writer = csv.writer(file)
        writer.writerow(value)

stemoscope_Hfeatures = pd.read_csv('stemoscope_Hfeatures.csv')
print ('stemoscope_Hfeatures\n', stemoscope_Hfeatures)

```

source2 LowerChestL.wav

label not_covid

source2 LowerChestLeft.wav

label not_covid

stemoscope_Hfeatures

	filename	chroma_stft	rmse	spectral_centroid	\
0	LowerChestL.wav	0.710613	0.007436	154.827284	
1	LowerChestLeft.wav	0.680129	0.006454	185.704678	

	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mfcc2	\
0	286.118120	222.709147	0.006031	-668.389587	133.516434	
1	304.865986	298.872884	0.007263	-642.639587	140.815155	

	mfcc3	...	mfcc12	mfcc13	mfcc14	mfcc15	mfcc16	\
0	90.437553	...	3.179373	5.708297	10.572505	12.559539	9.553345	
1	81.704582	...	12.340603	11.032862	7.448788	5.355366	5.225950	

	mfcc17	mfcc18	mfcc19	mfcc20	label
0	4.004292	0.11485	-0.069424	2.271688	not_covid
1	4.916399	3.40168	1.835933	1.360063	not_covid

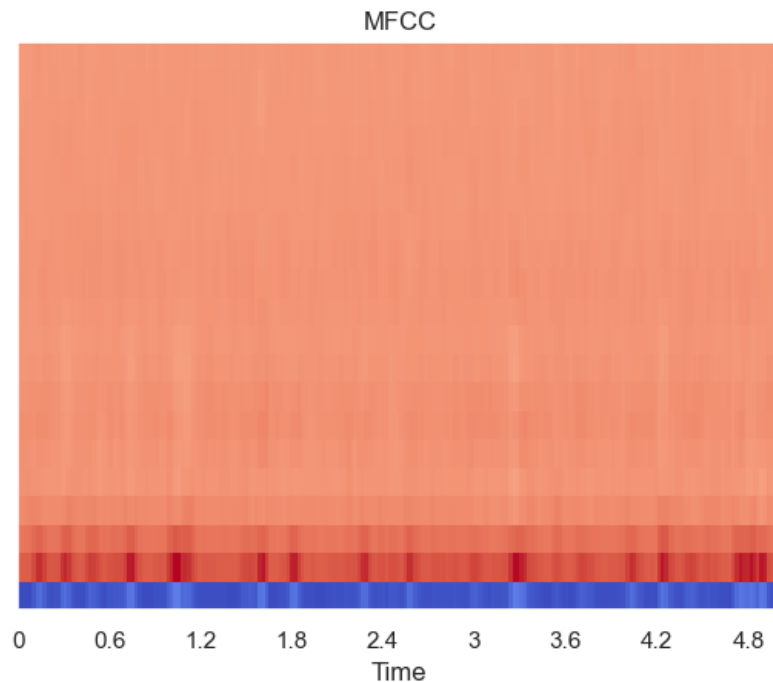


Fig 25. Spectrogram of Heart Audio

```
In [62]: # Drawing mel spectrogram
y,sr = librosa.load('Heart_Audio/LowerChestLeft.wav', mono=True)
fig, ax = plt.subplots(sharex=True, sharey=True)
#draw sound with time axis
librosa.display.waveplot(y, sr=sr, max_points=50000.0, x_axis='time')

S = librosa.feature.melspectrogram(y=y, sr=sr, n_mels=129, fmax=8000)

print ('S.shape',S.shape)
fig, ax = plt.subplots()
S_dB = librosa.power_to_db(S, ref=np.max)
img = librosa.display.specshow(S_dB, x_axis='time',
                               y_axis='mel', sr=sr,
                               fmax=8000, ax=ax)
fig.colorbar(img, ax=ax, format='%+2.0f dB')
ax.set(title='Mel-frequency spectrogram 2')

S.shape (129, 1218)
```

```
Out[62]: [Text(0.5, 1.0, 'Mel-frequency spectrogram 2')]
```

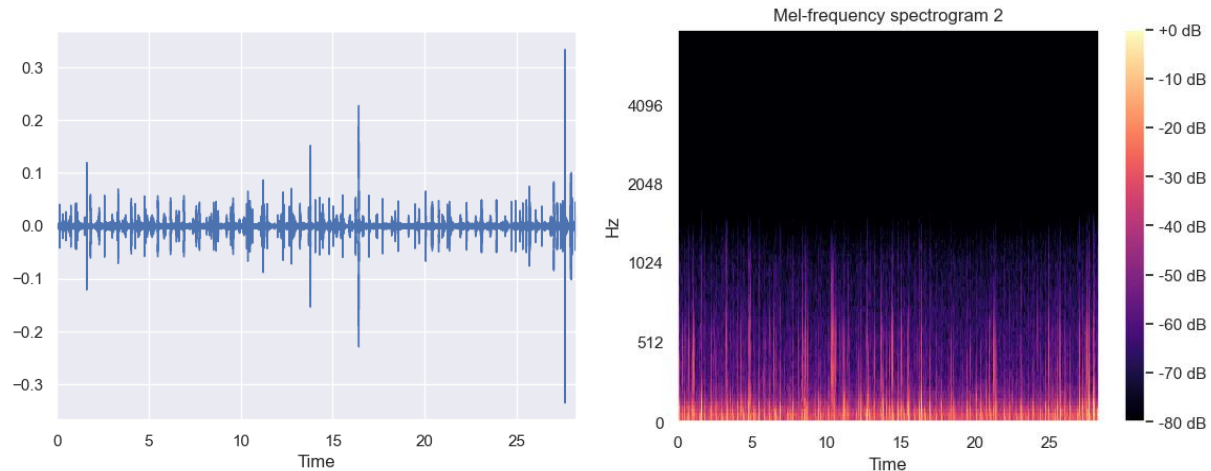



Fig 26 Mel Spectrogram of Heart Audio

Spectrograms of Lung Sounds in Training datasets

Extracting the Spectrogram for every Audio File

```
In [38]: #Loading csv file
train_csv = pd.read_csv("cough_trial_extended.csv") #Read csv.file
dataset = "cough_trial_extended.csv" # Path
anothercsv = pd.read_csv('cough_dataset.csv')
print (train_csv.head(4)) # Get the first 4 values
print (anothercsv)
```

	file_properties	class
0	0v8MGxNetjg_ 10.000_ 20.000.wav	not_covid
1	1j1duoxdxBg_ 70.000_ 80.000.wav	not_covid
2	1MSY04wgiag_ 120.000_ 130.000.wav	not_covid
3	1PajbAKd8Kg_ 0.000_ 10.000.wav	not_covid

	file	status
0	cough-shallow-3CwioNQVDBQ6CttLyFVRJpMpVhK2.wav	covid
1	pos-0421-084-cough-m-50.wav	covid
2	cough-heavy-6T43bddKokfG7MwnJWvrPZSsyrC2.wav	covid
3	cough-heavy-hNAGUEhL2Nh7V89at3yFEjQYo6c2.wav	covid
4	cough-heavy-hte8VptUoGVFEqvHpbh5brgfcNP2.wav	covid
..
745	6lbkx_tf50g_ 220.000_ 230.000.wav	not_covid
746	KoUw2T0QKGY_ 70.000_ 80.000.wav	not_covid
747	X0kpF1at8lM_ 0.000_ 10.000.wav	not_covid
748	j7MwYU5VH1s_ 70.000_ 80.000.wav	not_covid
749	o-TJISpYLFc_ 50.000_ 60.000.wav	not_covid

[750 rows x 2 columns]

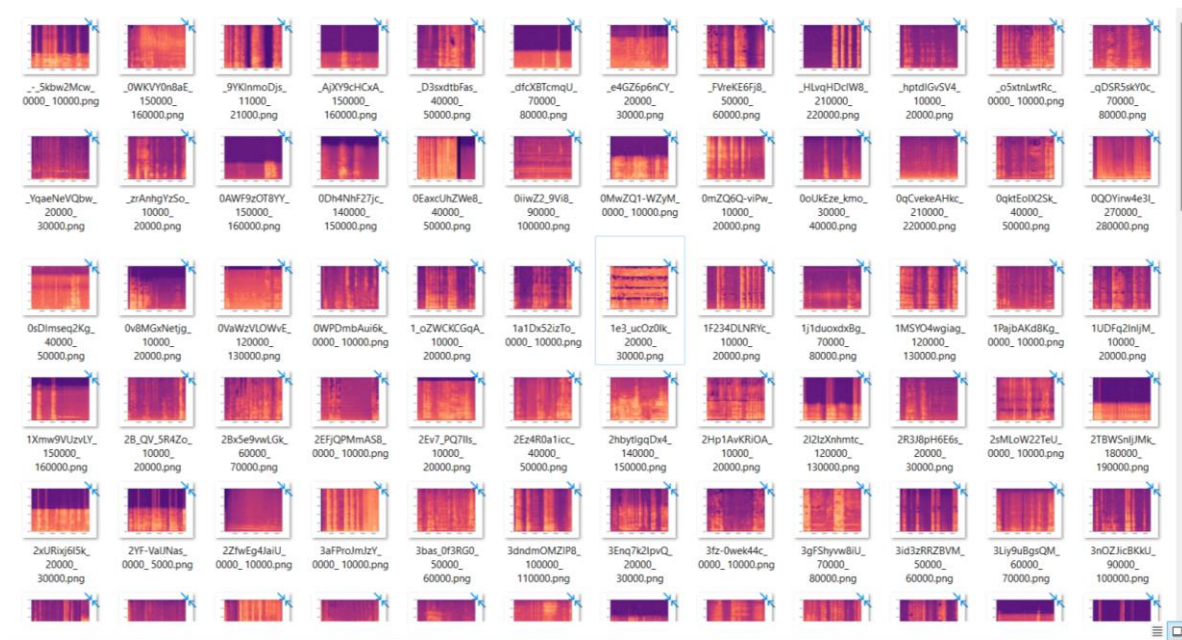


Fig 27 Mel Spectrogram of Training Lung Audio

The ANN Model used for Experimentation

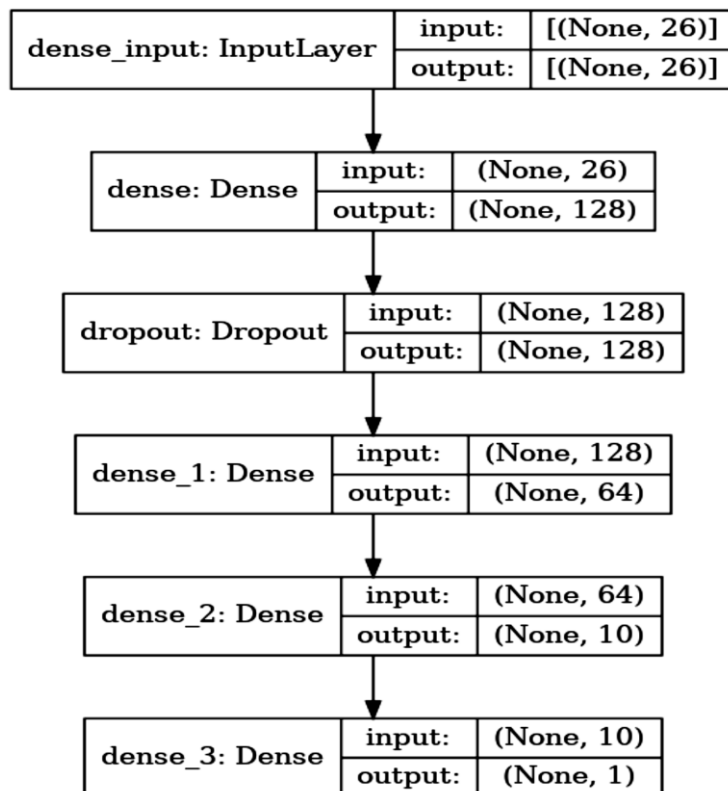


Fig 28 ANN Network

- The first layer is the input layer. Following which is the first dense layer with 128 neurons.
- The third layer is a dropout layer to counter overfitting
- Following this, to ensure that the data is deciphered better, there are three more dense layers comprising 128, 64 and 10 neurons each.

Activation Function:

$$S(x) = \frac{1}{1 + e^{-x}}$$

- For the sigmoid function, that results in a value between 0 and 1, **0.5** was taken as the threshold value.
- For all dense and dropout layers – ReLU Activation function was used.

Hyperparameter Tuning

Parameter	Value
Dropout Rate	0.3
Activations in Hidden Layers	ReLU
Activation in Output Layer	Sigmoid
Number of Epochs	15
Optimizer	Adam
Loss	'binary_crossentropy'
Metrics	'accuracy'

```
In [15]: def get_model():
model = models.Sequential()
model.add(layers.Dense(512, activation='relu', input_shape=(X_train.shape[1],))) #Input has been transpose

model.add(layers.Dense(256, activation='relu'))

model.add(Dropout(0.2))

model.add(layers.Dense(128, activation='relu'))

model.add(layers.Dense(64, activation='relu'))

model.add(Dropout(0.2))

model.add(layers.Dense(10, activation='relu'))

model.add(layers.Dense(2, activation='softmax'))

model.compile(optimizer= keras.optimizers.Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

    return model
#This model has signs of over fitting so let it drop out

# plot model
model = get_model()
model.summary()

Model: "sequential"
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	78848
dense_1 (Dense)	(None, 256)	131328
dropout (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
dense_3 (Dense)	(None, 64)	8256
dropout_1 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 10)	650
dense_5 (Dense)	(None, 2)	22
Total params: 252,000		
Trainable params: 252,000		
Non-trainable params: 0		

Results

Evaluation of Model

- Model performance is evaluated for accuracy, F-1 score, compared with some recorded respiratory sound signals.

- And different ML-Techniques is compared along with the previous research done.

```
In [16]: batch_size = 16
early_stopping_patience = 10

# Add early stopping

my_callbacks = [
    tf.keras.callbacks.ModelCheckpoint(filepath='./model_{epoch:02d}.h5',
                                       save_freq='epoch',
                                       save_best_only=True,
                                       period = 10),
    tf.keras.callbacks.EarlyStopping(
        monitor="val_loss", patience=early_stopping_patience, restore_best_weights=True
    )
]

history = model.fit(X_train, y_train,
                   epochs=100,
                   batch_size=batch_size,
                   callbacks = my_callbacks,
                   validation_split=0.15)
```

WARNING:tensorflow:'period' argument is deprecated. Please use 'save_freq' to specify the frequency in number of batches seen.

Epoch 1/100
173/173 [=====] - 1s 4ms/step - loss: 0.4091 - accuracy: 0.8383 - val_loss: 0.3664 - val_accuracy: 0.8507
Epoch 2/100
173/173 [=====] - 0s 3ms/step - loss: 0.3188 - accuracy: 0.8738 - val_loss: 0.3712 - val_accuracy: 0.8630
Epoch 3/100
173/173 [=====] - 0s 2ms/step - loss: 0.2708 - accuracy: 0.8966 - val_loss: 0.3476 - val_accuracy: 0.8485
Epoch 4/100
173/173 [=====] - 0s 2ms/step - loss: 0.2353 - accuracy: 0.9089 - val_loss: 0.3129 - val_accuracy: 0.8589
Epoch 5/100

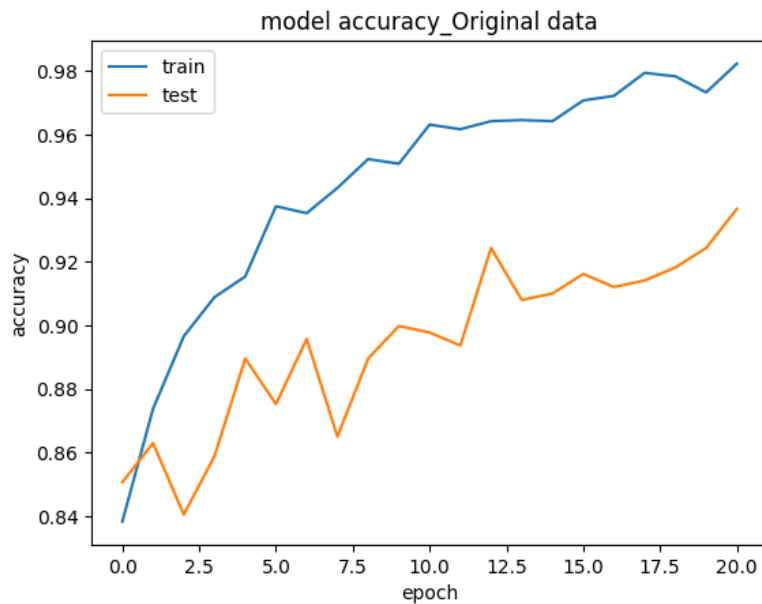


Fig 29 Accuracy

```
In [17]: def history_loss_acc(history,name):
# list all data in history
print(history.history.keys())

# summarize history for accuracy

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy_'+name)
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss_'+name)
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

history_loss_acc(history, 'Original data')

dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

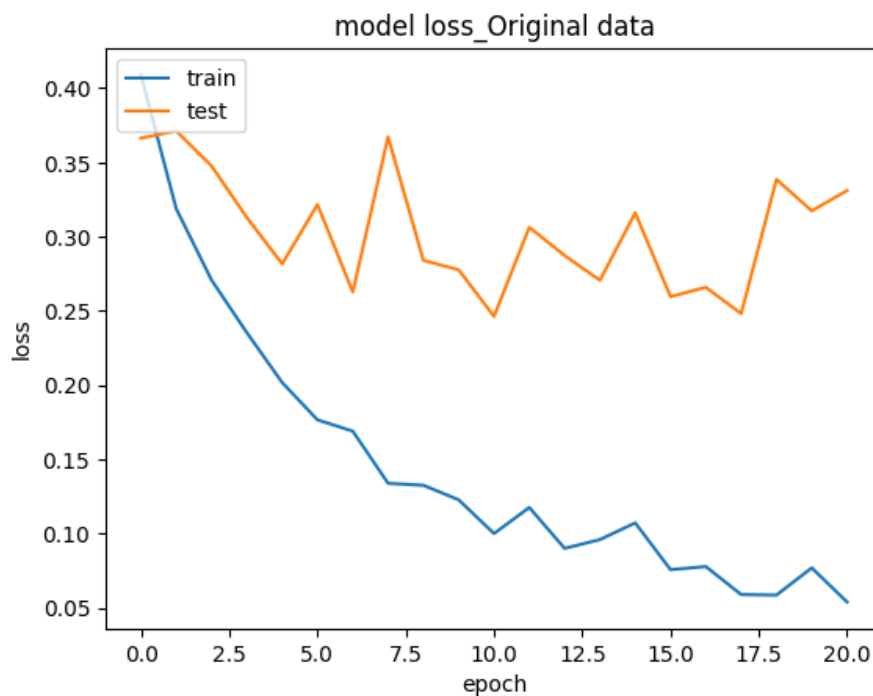


Fig 30 Model Loss

Predictions on Test Data

```
In [19]: test_loss, test_acc = model.evaluate(X_test,y_test)
```

```
26/26 [=====] - 0s 1ms/step - loss: 0.2431 - accuracy: 0.8956
```

```
In [20]: print('test_acc: ',test_acc)
```

```
test_acc: 0.8955773711204529
```

```
2]: ## predictions = np.array([1 if x >= 0.5 else 0 for x in seed_final_test])
def evaluate_matrix(y_test, y_predict, name):
    cm = confusion_matrix(y_test, y_predict)
    cm_df = pd.DataFrame(cm, index=["Negative", "Positive"], columns=["Negative", "Positive"])

    plt.figure(figsize=(10, 10))

    sns.set(font_scale=1)

    ax = sns.heatmap(cm_df, annot=True, square=True, fmt='d', linewidths=.2, cbar=0, cmap=plt.cm.Blues)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=0)

    plt.ylabel("True labels")
    plt.xlabel("Predicted labels")
    plt.tight_layout()
    plt.title(name)

    plt.show()

    print(classification_report(y_test, y_predict, target_names=["Negative", "Positive"]))

evaluate_matrix(y_test, y_predict, 'Original model')
```

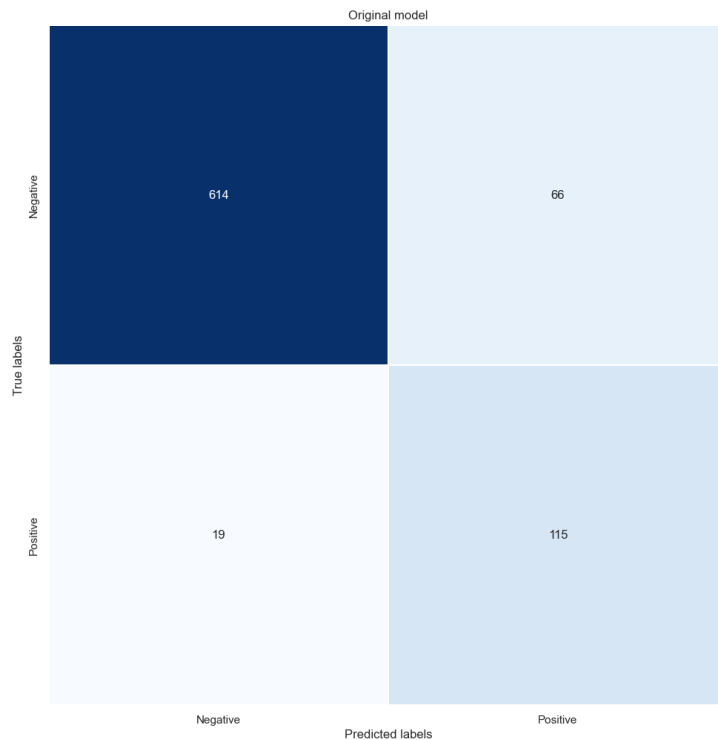


Fig 30 Prediction Heatmap

ROC_curve

```
In [23]: # summarize score
# print(predictions[:,1], '\n', predictions[:,1].shape )
def ROC_curve(y_test, predictions, name):

    # calculate roc curves
    lr_fpr, lr_tpr, _ = roc_curve(y_test, predictions[:,1])
    print ('model: {} \nAUC = {}'.format(name, auc(lr_fpr, lr_tpr)))
    # plot the roc curve for the model
    lw = 2
    plt.plot(lr_fpr, lr_tpr, color="darkorange",
             lw=lw, label="ROC curve (area = %0.2f)" % auc(lr_fpr, lr_tpr))
    plt.plot([0, 1], [0, 1], color="navy", lw=lw, linestyle="--")
    plt.xlim([-0.02, 1.0])
    plt.ylim([0.0, 1.05])
    # axis labels
    pyplot.xlabel('False Positive Rate')
    pyplot.ylabel('True Positive Rate')
    plt.title(name)
    # show the legend
    pyplot.legend()
    # show the plot
    pyplot.show()
```

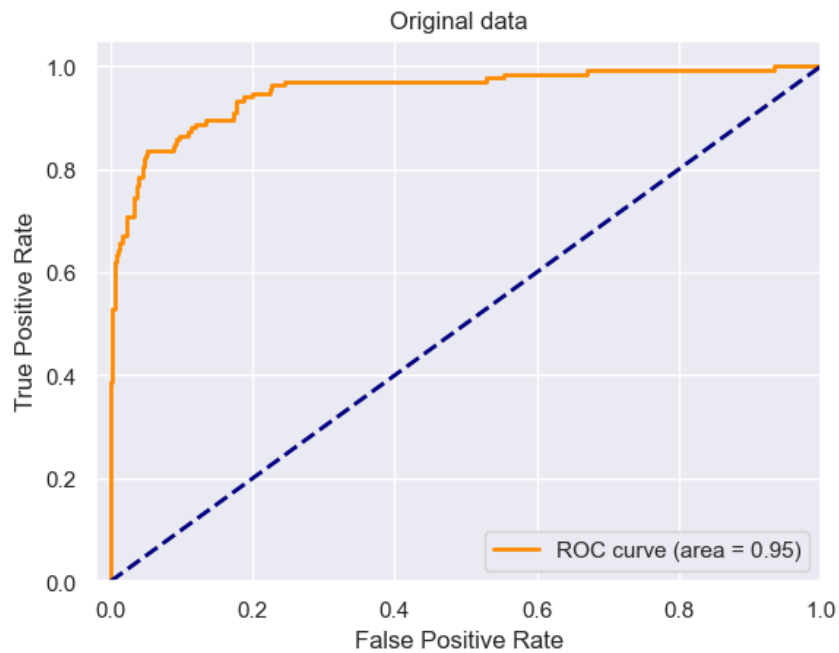


Fig 31 ROC Curve

	precision	recall	f1-score	support
Negative	0.97	0.90	0.94	680
Positive	0.64	0.86	0.73	134
accuracy			0.90	814
macro avg	0.80	0.88	0.83	814
weighted avg	0.91	0.90	0.90	814

Treating Imbalanced data

- The Dataset used, in this project is imbalanced positive: 669 (16.45% of total), negative cases: 3399.
- We can use oversampling with SMOTE (Synthetic Minority Oversampling Technique)
- SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors.
- The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space.

```
# transform the dataset
oversample = SMOTE(sampling_strategy=0.5, k_neighbors=5) #pos is equal to 50% neg
X_os, y_os = oversample.fit_resample(X_train, y_train)

order = np.arange(len(y_os))
np.random.shuffle(order)
X_os = X_os[order]
y_os = y_os[order]

neg_os, pos_os = np.bincount(y_os)
total_os = neg_os + pos_os
print('\nAfter oversampling \nnegative cases: {} \npositive cases: {} ({:.2f}% of total)'.format(neg_os,
```

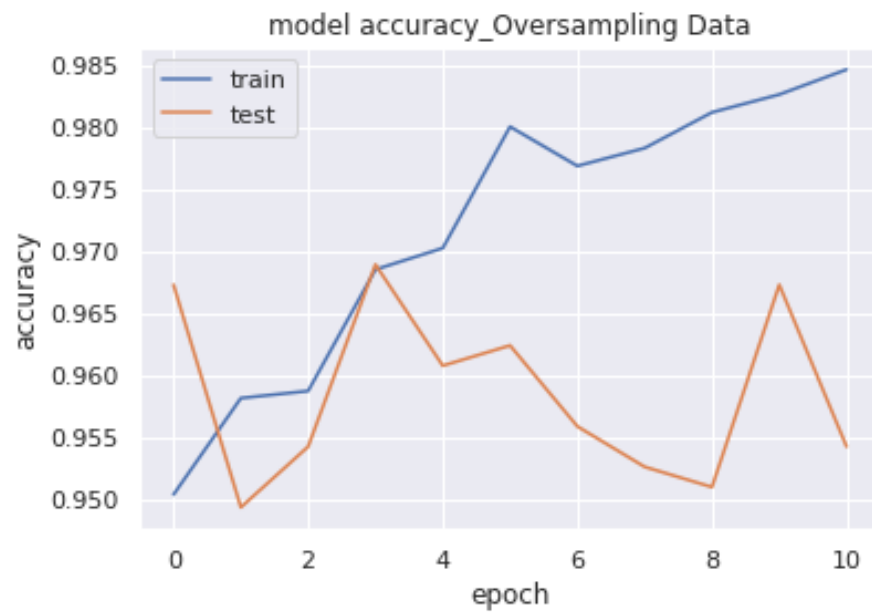


Fig 32 Oversampling

	precision	recall	f1-score	support
Negative	0.97	0.97	0.97	680
Positive	0.86	0.86	0.86	134
accuracy			0.95	814
macro avg	0.92	0.92	0.92	814
weighted avg	0.95	0.95	0.95	814

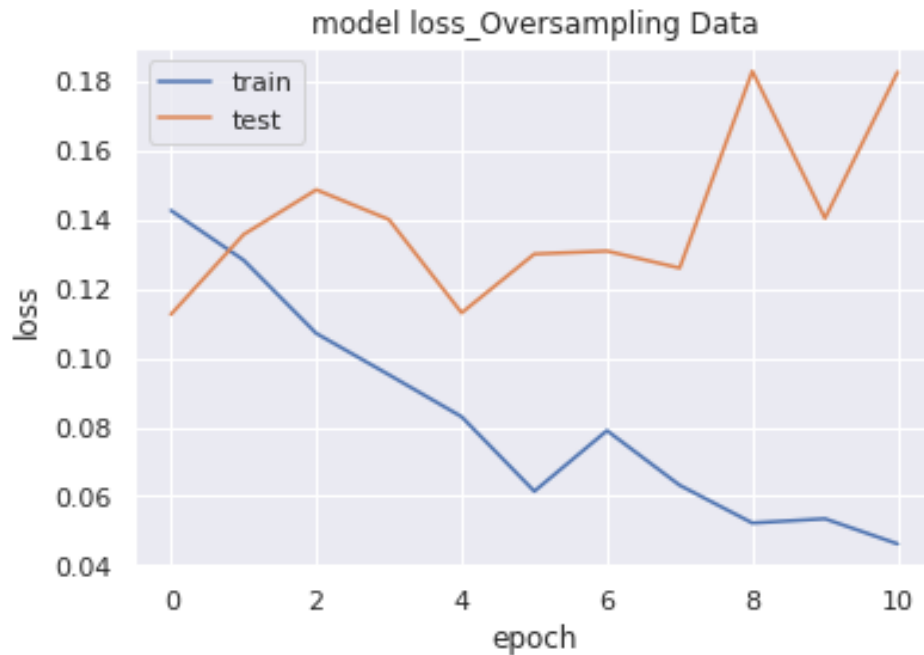


Fig 33. Loss over Oversampling

Comparing with LightGBM

```

kf_x_train, kf_y_train = Xs[trn_idx], ys[trn_idx]
kf_x_valid, kf_y_valid = Xs[val_idx], ys[val_idx]

print(kf_x_train.shape, kf_y_train.shape)

dtrain = lgb.Dataset(kf_x_train, kf_y_train)
dvalid = lgb.Dataset(kf_x_valid, kf_y_valid)

params["random_state"] = seed
params["num_leaves"] = 500 + seed * 100

model = lgb.train(params,
                  dtrain,
                  num_boost_round=NUM_BOOST_ROUND,
                  early_stopping_rounds=params["early_stopping_rounds"],
                  valid_sets=(dtrain, dvalid),
                  valid_names=("train", "valid"),
                  verbose_eval = 100)

```

	precision	recall	f1-score	support
Negative	0.97	0.97	0.97	664
Positive	0.84	0.85	0.84	136
accuracy			0.95	800
macro avg	0.90	0.91	0.91	800
weighted avg	0.95	0.95	0.95	800

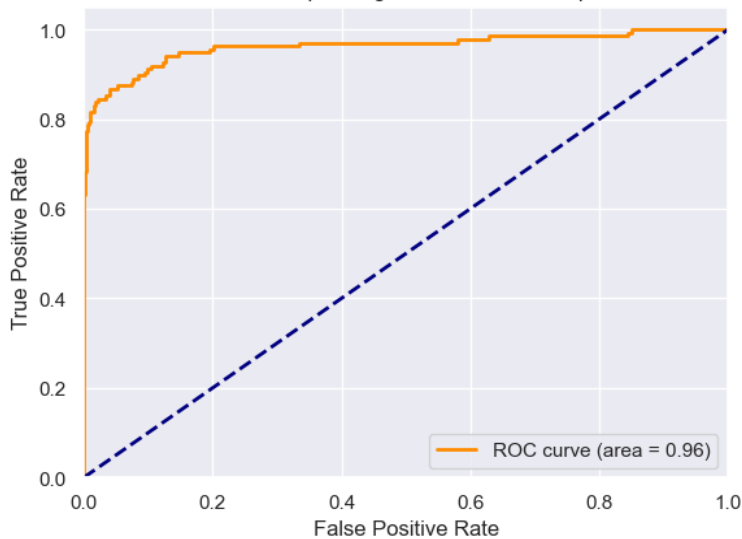


Fig 34. ROC for Oversampling

Conclusion

The project goes through some COVID-19 cough classifiers using smartphone-based sensor audio recordings and different machine learning models. To train and evaluate these models, the project uses the Coswara dataset combined with some other datasets. Although the systems addressed by the project describe require more stringent validation on a larger dataset, the results presented are very promising and indicate that COVID-19 screening based on automatic classification of coughing sounds is viable. Since the data has been captured on smartphones-based sensors, and since the classifier can in principle also be implemented on such device, such cough classification is cost-efficient, easy to apply and deploy. Furthermore, it could be applied remotely, thus avoiding contact with medical personnel.

Future Scope

The project has addressed machine learning models for classifying Covid-19 positive and healthy samples. Further the dataset can be used to train CNN, DNN, ANN models such as Resnet50, VGG16 to get more accurate results. Also, Hyperparameter tuning remains an important aspect while designing these models which can be implemented to get better results.

References

- [1]. Zhiang Chen, Muyun Li, Ruoyu Wang, Wenzhuo Sun, Jiayi Liu, Haiyang Li, Tianxin Wang, Yuan Lian, Jia-qian Zhang, Xinheng Wang, "Diagnosis of COVID-19 via acoustic analysis and artificial intelligence by monitoring breath sounds on smartphones," *Journal of Biomedical Informatics*, Volume 130, 2022, 104078, ISSN 1532-0464, <https://doi.org/10.1016/j.jbi.2022.104078>.
- [2]. Vincenzo Dentamaro, Paolo Giglio, Donato Impedovo, Luigi Moretti, Giuseppe Pirlo, "AUCO ResNet: an end-to-end network for COVID-19 pre-screening from cough and breath," *Pattern Recognition*, Volume 127, 2022, 108656, ISSN 0031-3203, <https://doi.org/10.1016/j.patcog.2022.108656>.
- [3]. Son, Myoung-Jin, and Seok-Pil Lee. "COVID-19 Diagnosis from Crowdsourced Cough Sound Data." *Applied Sciences* 12, no. 4 (2022): 1795.
- [4]. Abdel-Jaber, Hussein, Disha Devassy, Azhar Al Salam, Lamy Hidaytallah, and Malak EL-Amir. "A Review of Deep Learning Algorithms and Their Applications in Healthcare." *Algorithms* 15, no. 2 (2022): 71.
- [5]. Darici, Esin, Nicholas Rasmussen, Jaclyn Xiao, Gunvant Chaudhari, Akanksha Rajput, Praveen Govindan, Minami Yamaura, Laura Gomezjurado, Amil Khanzada, and Mert Pilanci. "Using Deep Learning with Large Aggregated Datasets for COVID-19 Classification from Cough." *arXiv preprint arXiv:2201.01669* (2022).
- [6]. Elena A. Lapteva, Olga N. Kharevich, Victoria V. Khatsko, Natalia A. Voronova, Maksim V. Chamko, Irina V. Bezruchko, Elena I. Katibnikova, Elena I. Loban, Mostafa M. Mouawie, Helena Binetskaya, Sergey Aleshkevich, Aleksey Karankevich, Vitaly Dubinetski, Jørgen Vestbo, Alexander G. Mathioudakis. "Automated lung sound analysis using the LungPass platform: a sensitive and specific tool for identifying lower respiratory tract involvement in COVID-19". *EurRespir J* 2021; in press (<https://doi.org/10.1183/13993003.01907-2021>).
- [7]. J. R. Balbin, A. I. T. Yap, B. D. Calicdan and L. A. M. Bernabe, "Arrhythmia Detection using Electrocardiogram and Phonocardiogram Pattern using Integrated Signal Processing Algorithms with the Aid of Convolutional Neural Networks," 2021 IEEE International Conference on Automatic Control & Intelligent Systems (I2CACIS), 2021, pp. 146-151, doi: 10.1109/I2CACIS52118.2021.9495913.
- [8]. Sharma, Neeraj, Prashant Krishnan, Rohit Kumar, Shreyas Ramoji, Srikanth Raj Chetupalli, Prasanta Kumar Ghosh, and Sriram Ganapathy. "Coswara--a database of breathing, cough, and voice sounds for COVID-19 diagnosis." *arXiv preprint arXiv:2005.10548* (2020).

- [9]. Santosh, K. C., Nicholas Rasmussen, Muntasir Mamun, and Sunil Aryal. "A systematic review on cough sound analysis for COVID-19 diagnosis and screening, Is my cough sound COVID-19." *PeerJ Computer Science* 8 (2022): e958.
- [10]. Eni M, Mordoh V, Zigel Y (2022) Cough detection using a non-contact microphone: A nocturnal cough study. *PLOS ONE* 17(1): e0262240. <https://doi.org/10.1371/journal.pone.0262240>.
- [11]. Dunne, Rob, Tim Morris, and Simon Harper. "High accuracy classification of COVID-19 coughs using Mel-frequency cepstral coefficients and a convolutional neural network with a use case for smart home devices." (2020).
- [12]. Chaudhari, G., "Virufy: Global Applicability of Crowdsourced and Clinical Datasets for AI Detection of COVID-19 from Cough", arXiv e-prints, 2020.
- [13]. A systematic review on cough sound analysis for Covid-19 diagnosis and screening: is my cough sound COVID-19? <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9138020/>

Other References:

- ❖ Audio Deep Learning Made Simple: Sound Classification, Step-by-Step
<https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5>
- ❖ Machine Learning for Audio Classification
<https://www.section.io/engineering-education/machine-learning-for-audio-classification/>
- ❖ Artificial intelligence model detects asymptomatic Covid-19 infections through cellphone recorded coughs
<https://news.mit.edu/2020/covid-19-cough-cellphone-detection-1029>
- ❖ AI can detect COVID-19 from the sound of your cough
<https://www.livescience.com/asymptomatic-coronavirus-detection-ai.html>

Software Packages:

- ❖ Tensor flow-based Audio-Classfier API on Anaconda packages.
https://www.tensorflow.org/lite/inference_with_metadata/task_library/audio_classifier
- ❖ Hugging face-Transformers based Audio classifiers.
https://huggingface.co/docs/transformers/tasks/audio_classification
- ❖ Fastaudio notebooks.
<https://cran.r-project.org/web/packages/fastai/vignettes/audio.html>
<https://fastaudio.github.io/Introduction%20to%20Fastaudio/>
- ❖ Python package to extract the features from any audio dataset.
<https://pypi.org/project/audio-classification-features/>
- ❖ MATLAB based Audio classifier tools

<https://www.mathworks.com/help/audio/gs/classify-sound-using-deep-learning.html>

- ❖ Urban Sound Classification Using Deep Learning

https://www.mathworks.com/matlabcentral/fileexchange/96148-urban-sound-classification-using-deep-learning?s_tid=FX_rc2_behav