

COVID-19 AI DIAGNOSIS USING COUGH RECORDINGS

COVID-19 AI DIAGNOSIS USING COUGH RECORDINGS
A MINI PROJECT REPORT

Submitted by

ANEESA.K

S.DHIVYA

KAAVYA.L

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

EASWARI ENGINEERING COLLEGE, CHENNAI

ANNA UNIVERSITY:: CHENNAI 600 025

JULY 2021

APPENDIX 2

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this Mini project report “**COVID-19 AI DIAGNOSIS USING COUGH RECORDINGS**”

is the bonafide work of

ANEESA K

DHIVYA S,

KAAVYA L

who carried out the project work under my supervision.

SIGNATURE

Dr.K.M. ANANDKUMAR

(HEAD OF THE DEPARTMENT)

Computer Science and Engineering

Easwari Engineering College
Bharathi Salai,Ramapuram,
Chennai 600 089

SIGNATURE

Mr. K.P.K DEVAN

(SUPERVISOR)

Assoc. Professor

Computer Science and Engineering

Easwari Engineering College
Bharathi Salai, Ramapuram,
Chennai 600 089

ACKNOWLEDGEMENT

We hereby place our deep sense of gratitude to our beloved Founder Chairman of the institution, **Dr.T.R.Pachamuthu, B.Sc., M.I.E.**, for providing us with the requisite infrastructure throughout the course.

We would also like to express our gratitude towards our Chairman **Dr.R.Shivakumar, M.D., Ph.D.** for giving the necessary facilities.

We convey our sincere thanks to **Dr.R.S.Kumar, M.Tech., Ph.D.** Principal Easwari Engineering College, for his encouragement and support. We extend our hearty thanks to **Dr.V.Elango, M.E., Ph.D.**, Vice Principal (academics) and **Dr.K.M.Anandkumar, M.Tech, Ph.D**, Vice Principal (admin), Easwari Engineering College, for their constant encouragement.

We take the privilege to extend our hearty thanks to **Dr.K.M.Anand Kumar, M.Tech, Ph.D.**, Head of the Department, Computer Science and Engineering, Easwari Engineering College for his suggestions, support and encouragement towards the completion of the project with perfection.

We would also like to express our gratitude to our guide **Mr. K.P.K Devan, Assoc. Professor, M.Tech, (Ph.D)** Department of Computer Science and Engineering, Easwari Engineering College, for his/her constant support and encouragement.

Finally, we wholeheartedly thank all the faculty members of the Department of Computer Science and Engineering for warm cooperation and encouragement.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
	Abstract	ii
1	Introduction	3
	1.1. General	3
	1.2. Problem Description	3
	1.3. Objective	4
	1.4. Scope of Project	4
	1.5. Organization of the Project	5
2	Literature Survey	6
	2.1. General	6
	2.2. Existing Systems	6
	2.3. Issues in Existing Systems	8
	2.4. Proposed System	8
	2.5. Summary	8
3	System Design	9
	3.1. General	9
	3.2. System Architecture	9
	3.3. Functional Architecture	11
	3.4. Modular Design	12
	3.5. System Requirements	14
	3.6. Summary	14
4	System Implementation	15
	4.1. General	15
	4.2. Overview of the platform	15
	4.3. Module Implementation	19
	4.4. Summary	23
5	System Testing and Performance Analysis	24
	5.1. General	24
	5.2. Test Cases	25
	5.3. Performance Measures	25
	5.4. Performance Analysis	25
	5.5. Summary	28
6	Conclusion and Future Works	29
	Appendices	29
	Reference	41

ABSTRACT

Fast diagnosis of COVID-19 is important in stopping the spread of the epidemic. Using machine learning techniques, we may be able to recognize COVID-19 status through cough recordings. We show here that a CNN using audio converted to Mel-frequency cepstral coefficient spectrogram images as input can achieve high accuracy results; with classification of validation data scoring an accuracy of 98% correct classification of COVID and not COVID labelled audio. The results are highly encouraging and provide further opportunities for research by the academic community on this important topic.

AI techniques can produce a free, non -Invasive, real-time, any-time, instantly distributable, large scale COVID-19 asymptomatic screening tool to augment current approaches in containing the spread of COVID-19. Practical use cases could be for daily screening of students, workers, and public as schools, jobs, and transport reopen, or for pool testing to quickly alert of outbreaks in groups.

CHAPTER 1

INTRODUCTION

1.1 GENERAL

There have been over 120 million people infected by COVID-19. Fast diagnosis will be extremely helpful for stopping the spread of pandemic areas. Cough is a common symptom of COVID-19.

Machine learning turns out to be a good approach for diagnosing COVID-19 using cough audios. Previous studies have shown it promising to automatically diagnose COVID-19 through cough and other audio data.

1.2 PROBLEM DESCRIPTION

1. The limitations of current viral and serology tests and the lack of complementary pre-screening methods to efficiently select who should be tested. They are expensive making the cost of testing a whole country each day impossible.
2. The “Trace, Test and Treat” strategy succeeded in flattening the pandemic curve (e.g., in South Korea, China and Singapore) in its early stages. However, in many parts of the world the pandemic has already spread to an extent that this strategy is not proving effective anymore.
3. The turnaround time for current tests is several days, recently stretching to 10 days in some countries as labs are becoming overwhelmed. By the time a patient is diagnosed using current methods, the virus has already been passed to many.
4. The in-person testing methods put the medical staff, particularly those with limited protection, at serious risk of infection. The inability to protect our medics can lead to further shortage of medical care and increased distress on the already stressed medical staff.

1.3 OBJECTIVES

The major objective of this project is to provide the following

- **EASE OF ACCESS:** It is easily accessible to the people all over the world even from a developing or underrated region just with the availability of internet to access the web site.
- **ZERO-COST DISEASE DETECTION:** Anyone with the stable internet can easily access the model, detect and diagnose the disease at zero cost effectively.
- **TIME EFFICIENCY:** In certain regions of developing countries and in underrated areas, the unavailability of well experienced medical practitioners and diagnosticians increases the time needed to diagnose and detect the disease and sometimes it even takes weeks to get the report. As the time increases, the risk of the patient eventually increases in the case of positive diagnosis. So this model overcomes this risk factor by providing the report in a fraction of second thereby leading to faster diagnosis.
- **HIGHER ACCURACY:** On training the neural net using lots of data, the model learns hidden patterns from the data and learns the hypothesis that maps the input to the output thereby making predictions with more than 98% accuracy.
- **PROVIDES ASSISTANCE TO MEDICAL PRACTITIONERS:** This model can even provide assistance to medical practitioners and diagnosticians in making more accurate clinical decisions thereby reducing the rate of medical error and improving the efficiency effectively.

1.4 SCOPE OF THE PROJECT

The scope of the project is to create a model for detecting COVID-19 for everyone including asymptomatic individuals using cough recordings.

1.5 ORGANIZATION OF THE PROJECT

The report consists of 6 chapters, the contents of which are described below: Chapter 1 is the introduction that explains the basic information of the system. Chapter 2 is the literature survey that elaborates on the research works on the existing systems. Chapter 3 describes the system design. Chapter 4 gives details regarding the system implementation. Chapter 5 describes the performance analysis of the proposed system. Chapter 6 provides the conclusion, which summarizes the efforts undertaken in the proposed system and states findings and shortcomings in the proposed system.

CHAPTER 2

LITERATURE SURVEY

2.1 GENERAL

The documentation of a comprehensive review of the publishers and unpublished work from secondary sources data in the areas of specific interest to the researcher is referred to as a literature survey. It is a survey of all the techniques that have been used to summarize and analyze user reviews thus far. There have been major publications focusing on identifying diseases using deep learning techniques, but the number of research papers focused on large-scale application is limited. Any of the scientific papers and studies mentioned in this section were used to improve our proposed method. The fundamental properties and shortcomings of existing devices are discussed.

2.2 EXISTING SYSTEM

The contributions of various scholars are studied for survey and analyzing the merits and demerits in order to enhance the consequences for making the system work better.

Abdelfattah Hassan [1] claims that the implementation of Artificial Intelligence (AI) has made a significant contribution to the digital health district by applying the basics of Automatic Speech Recognition (ASR) and deep learning algorithms. In this study, we highlight the importance of speech signal processing in the process of early screening and diagnosing the COVID-19 virus by utilizing the Recurrent Neural Network (RNN) and specifically its significant well-known architecture, the Long Short-Term Memory (LSTM) for analysing the acoustic features of cough, breathing, and voice of the patients. Our results show a low accuracy in the voice test compared to both coughing and breathing sound samples. Moreover, our results are preparatory, and there is a possibility to enhance the accuracy of the voice tests by expanding the data set and targeting a larger group of healthy and infected people.

Jiaxuan Guo [2] suggests that rapid diagnosis will be critical in halting the spread of pandemic regions. Cough is a frequent COVID-19 symptom. Using cough audios to diagnose COVID-19, machine learning proves to be a viable option. Previous research has demonstrated that

using cough and other audio data to automatically diagnose COVID-19 is a promising method. For audio processing, a spectrogram is widely employed. Another quantity, mel frequency cepstrum (MFC), is extensively utilised as a characteristic for speech detection in human sound. Mel-frequency cepstrum coefficients contain the information in MFC (MFCCs). In audio processing libraries, there have been standard techniques for determining the MFCCs of an audio script. In the project, they have mainly used the Coswara and Coughvid data, because they have the highest data volume among publicly available datasets.

Rob Dunne [3] shows that a CNN can obtain high accuracy results utilising audio transformed to Mel-frequency cepstral coefficient spectrogram pictures as input, with classification of validation data scoring an accuracy of 97.5 percent correct classification of covid and non covid labelled audio. The study presented here shows that great accuracy may be obtained with a little dataset, which could have a big influence in this field. The findings are quite encouraging, and they open up new avenues for academic research on this vital issue. In this study, we look at the possibility of using smart devices like Amazon's Alexa or Google Home to do high-accuracy audio classification of COVID-19 coughs as a potential diagnostic software application that might be used at home or at work.

Jordi Laguarda [4] hypothesized a way using Artificial Intelligence in which Intelligence based on a forced-cough cell phone recording COVID-19 subjects, also those who were asymptomatic, could be accurately distinguished. Between April and May 2020, they constructed a data collection pipeline of COVID-19 cough recordings through the website (open.sigma.mit.edu) and developed the largest audioCOVID-19 cough balanced dataset recorded to date with 5,320 subjects to train their MIT Open Voice model. The architecture's COVID-19 discrimination accuracy was substantially improved by using transfer learning to learn biomarker features on larger datasets, which had previously been successfully tested in our lab on Alzheimer's. The model achieves COVID-19 sensitivity of 98.5 percent and specificity of 94.2 percent when validated with subjects diagnosed using an official examination (AUC: 0.97). It has a sensitivity of 100 percent and a specificity of 83.2 percent in asymptomatic subjects.

Ali Imran [5] suggests that the inability to test at scale has become humanity's Achilles's heel in the ongoing war against the COVID-19 pandemic. A scalable screening tool would be a game changer. Building on the prior work on cough based diagnosis of respiratory diseases, we propose, develop and test an Artificial Intelligence (AI)-powered screening solution for COVID-19 infection that is deployable via a smartphone app. The app, named AI4COVID-19 records and sends three 3-second cough sounds to an AI engine running in the cloud, and returns a result within two minutes. Results show AI4COVID -19 can distinguish between COVID-19 coughs and several types of non-COVID19 coughs. The accuracy is promising enough to encourage a large-scale collection of labeled cough data to gauge the generalization capability of AI4COVID-19. AI4COVID-19 is not a clinical grade testing tool. Instead, it offers a screening tool deployable anytime, anywhere, by anyone. It can also be a clinical decision assistance tool used to channel clinical-testing and treatment to those who need it the most, thereby saving more lives.

2.3 ISSUES IN EXISTING SYSTEM

Many existing systems aim to diagnosis of COVID-19 using chest x-rays .There is no way of detecting COVID-19 in the comfort of our home. All the merits and demerits of some existing systems are taken into consideration to build the proposed system.

2.4 PROPOSED SYSTEM

The proposed system is to diagnose COVID-19 using only cough recordings that can be recorded through a cell phone. Using Convolutional Neural Network (CNN), we collect datasets of people's cough recording of both COVID-19 and non-COVID-19 subjects and use 70% of the data to train and 30% of the data to test. The spectrograms will be extracted from the recording data and then the features of the spectrogram will be extracted into Mel-frequency cepstral coefficients (MFCC)(20 in number), Spectral Centroid, Zero Crossing Rate, Chroma Frequencies, Spectral Roll-off. The model is then trained and makes predictions on the test data set for which get an accuracy of the system.

2.5 SUMMARY

The literature survey has covered information regarding existing systems. The issues and challenges have been identified in related work. This chapter has highlighted the proposed model.

CHAPTER 3

SYSTEM DESIGN

3.1 GENERAL

This chapter deals with the design aspect of the proposed system. Systems design implies a systematic approach to the design of a system. It may take a bottom- up or top-down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. Systems design then overlaps with systems analysis, systems engineering and systems architecture.

3.2 SYSTEM ARCHITECTURE

A system architecture, also known as a systems model, is a conceptual model that describes a system's structure, actions, and other aspects. A systematic explanation and representation of a system structured in a way that facilitates thinking about the system's structures and behaviors is called an architecture description. A system architecture may be made up of system modules and subsystems that will collaborate to execute the overall system. There have been attempts to formalize languages for describing machine architecture, which are referred to as Architecture Description Languages collectively (ADLs). This model has been designed with a basic and easy user interface in mind, so that people of any educational level can use it with ease. Users are required to provide their cough recordings to diagnose the disease. The deep learning model was developed and trained using datasets from the Kaggle website[6]. Figure 3.1 represents system architecture of the proposed system.

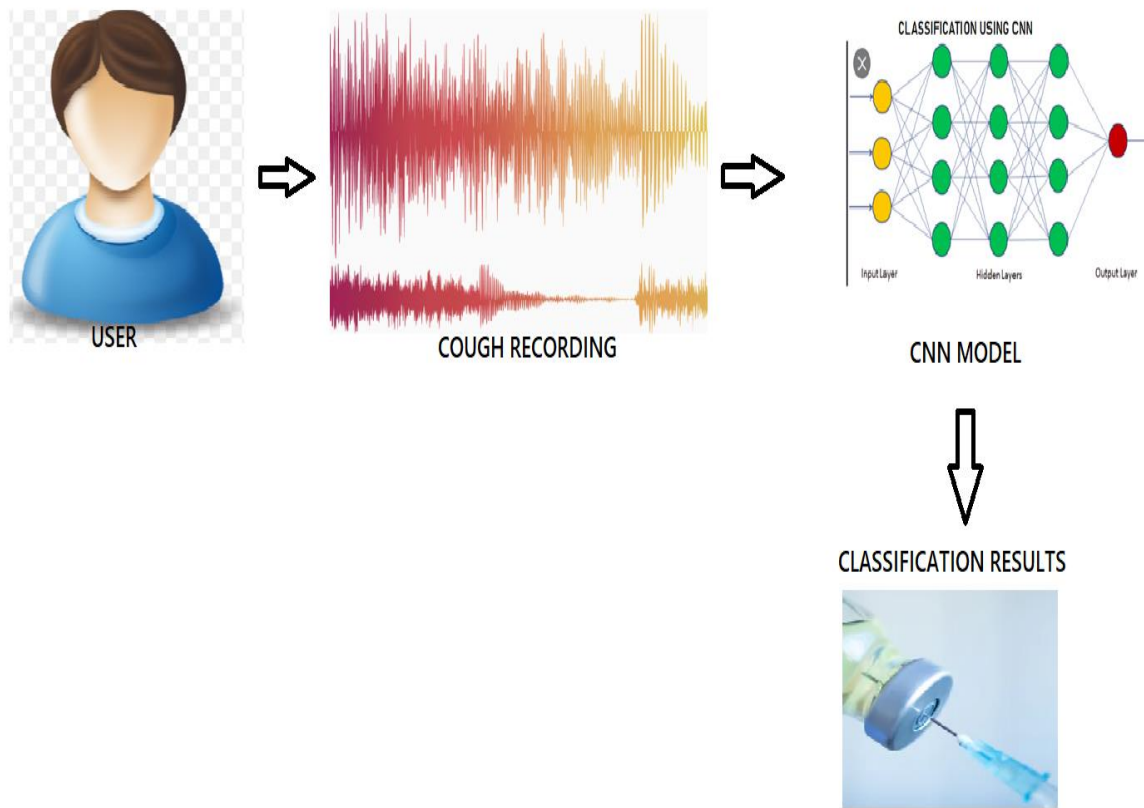


Figure 3.1: System Architecture

3.3 FUNCTIONAL ARCHITECTURE

A functional architecture is an architectural model that describes the functions of a system and how they interact. It specifies how the functions will work together to achieve the system's goal (s). In most cases, multiple architectures can meet the requirements. The cost, schedule, efficiency, and risk implications of each architecture and its collection of associated assigned specifications are usually different. The functional architecture is used to facilitate the construction of functional and performance tests. It also aids in the creation of verification tasks that are specified to verify the functional, performance, and constraint specifications, in conjunction with the physical architecture. The primary aim of the project is to predict if a person has COVID-19 using only cough recordings. Figure 3.2 depicts the function of the proposed system.

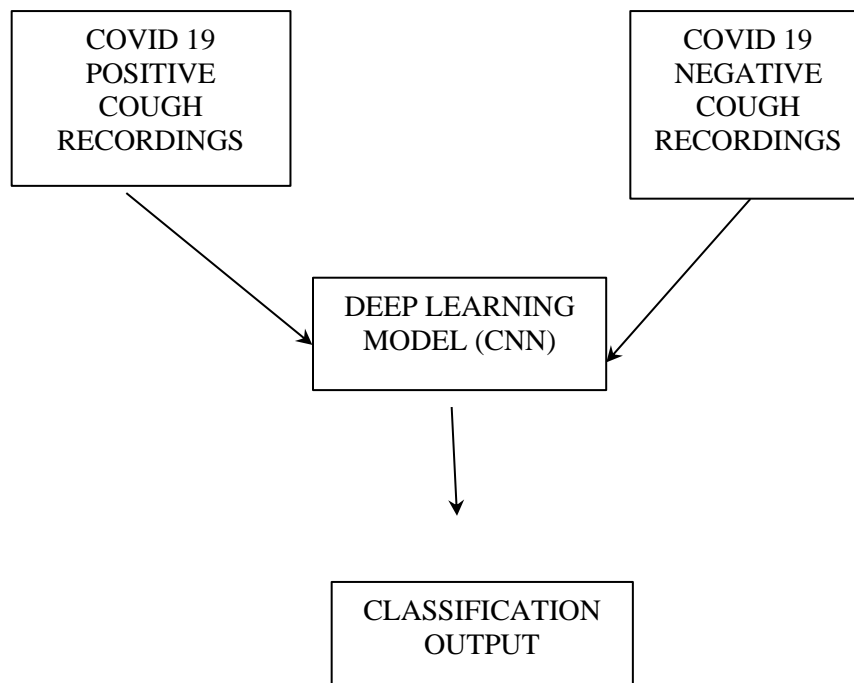


Figure 3.2: Functional Architecture

3.4 MODULAR DESIGN

Our approach consists of two principal steps. Extraction of the MFCC features, from the constant length examples and feeding them into an CNN model to make predictions. (line spacing?)

3.41. INPUT AUDIO SIGNAL

The training data used here are audio files from Kaggle[6]. We pre-process our audio data to extract the Melspectrogram. Mel-spectrogram is used as input of the model.

3.4.2. MEL-SPECTROGRAM

Mel scale is used to compute the mel-spectrogram. The Mel Spectrogram represents the time–frequency representation of the audio signal. Mel Spectrogram uses Mel filter and Mel Frequency to plot the signal. The Mel Filter is responsible for taking only those signals into consideration which falls under the range of Mel scale. Mel Filters function similar to human ears and the frequency at that range Mel scale is considered as Mel Frequency.

All the audio files get converted into their respective spectrograms. We can easily extract features from them.

Extracting features from Spectrogram

We will extract the following features:

- Mel-frequency cepstral coefficients (MFCC)(20 in number)
- Spectral Centroid,
- Zero Crossing Rate
- Chroma Frequencies
- Spectral Roll-off.

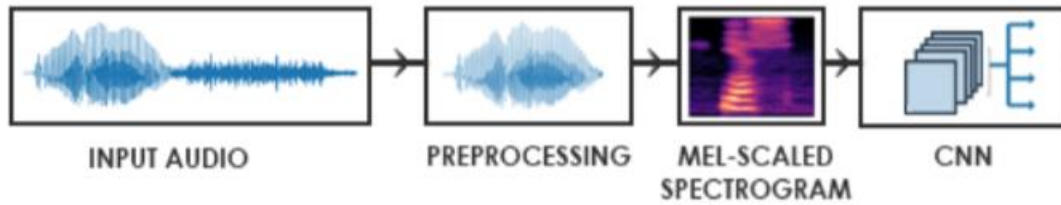


Figure 3.3 Audio Preprocessing

3.4.3 CONVOLUTIONAL NEURAL NETWORK

This step is a composition of feature reduction and classification. The Convolution Neural Network is responsible for the network layer operation. The convolution neural network is generally used in the classification of the image but in the proposed system. The audio signal behaves as the image on transforming audio signal into Mel-spectrogram.

The number of filters is 16. Batch size of 128. Number of epochs 100. We used `sparse_categorical_crossentropy` for loss function and Adam optimizer. Adambooster / optimizer?

Activation function: The **Rectified Linear Activation Function** or **ReLU** for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

Optimizer : Optimizer is used for fine-tuning weight values in a neural network. We use `sparse_categorical_crossentropy` as loss function and adam optimizer.

Metrics: Different types of metrics are used for the model evaluation. Such as accuracy, recall, precision, threshold etc.

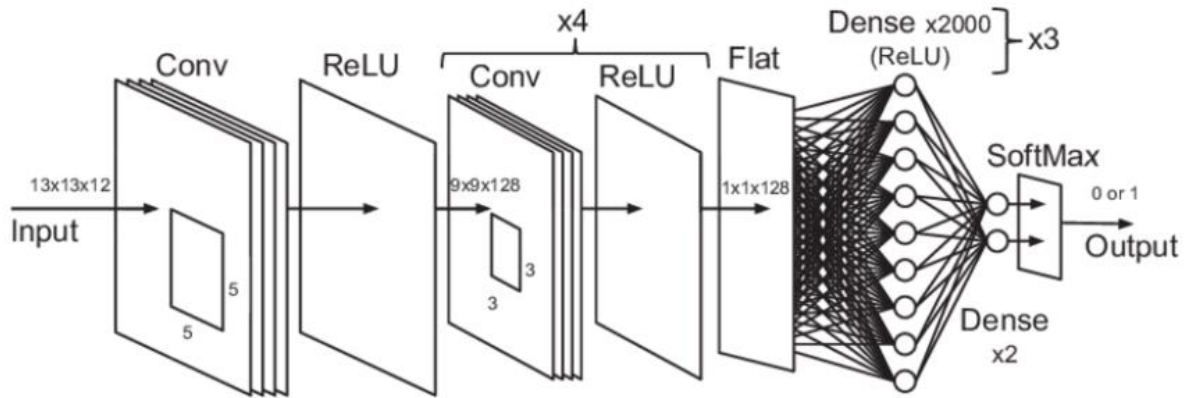


Figure 3.4 CNN Architecture

3.5. SYSTEM REQUIREMENTS

Hardware Specification

OS/ Platform : Since it is a cross-platform web application, it doesn't require a specific platform or an OS.

Software Specification

Language : Python (3.7+)

IDE : Jupyter Notebook/Google Collab

Libraries : TensorFlow, keras, librosa, matplotlib, Pandas and numpy.

3.6 SUMMARY

This chapter gives an overview of system design and its importance in the software life cycle. The functional architecture gives the entire functionality of the proposed system along with its modular structure and its interactions between modules.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 GENERAL

An implementation is the computer programming and deployment of a technical specification or algorithm as a program, software component, or other computer device. For a given specification or standard, there may be several implementations. High levels of user engagement and management support are usually beneficial to system implementation. Participation of users in the design and application of the system. Participation of users in the design and maintenance of information systems has a number of advantages. First, when consumers are heavily involved in system design, they gain more opportunities to shape the system to their priorities and business requirements, as well as more control over the result. Second, they are more likely to welcome change with open arms. Better solutions result from incorporating user experience and skills.

4.2 OVERVIEW OF THE SYSTEM

The following sections go through the various software and hardware requirements listed // check in the previous chapters. Before integrating them to incorporate the proposed framework, it's critical to understand how each variable works on its own.

4.2.1 PYTHON

Python is a high-level, interpreted programming language that can be used for a variety of tasks. Python was created by Guido van Rossum and first published in 1991. Its design style emphasizes code readability, with a lot of white space. It has constructs that allow for simple programming at both small and large scales. Van Rossum was the language community's chairman until July 2018, when he stepped down. Python has a dynamic style structure and memory management that is automated. It has a robust standard library and supports various programming paradigms, including object-oriented, imperative, functional, and procedural. For a wide range of operating systems, Python interpreters are available.

CPython, the standard Python implementation, is open source software with a community-based development model, as do virtually all of Python's other implementations. The Python Software Foundation, a non-profit organization, oversees Python and CPython. Python is designed to be a language that is simple to understand. Its formatting is clean and uncluttered, and it mostly uses English keywords instead of punctuation in other languages. It does not use curly brackets to delimit blocks, and semicolons after statements are optional, unlike many other languages. In comparison to C or Pascal, it has fewer syntactic exceptions and special cases.

4.2.2 TENSORFLOW

TensorFlow is a machine learning software library that is free and open-source. It can be used for a variety of activities, but it focuses on deep neural network training and inference. TensorFlow is a data-flow and differentiable programming-based symbolic math library. At Google, it's used for both research and development. The Google Brain team created TensorFlow for internal Google use. It was released in 2015 under the Apache License 2.0.

4.2.3 KERAS

Keras is an open-source software library for artificial neural networks that offers a Python interface. Keras serves as a user interface for TensorFlow. Keras supported a variety of backends up until version 2.3, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. Only TensorFlow is supported as of version 2.4. It is user-friendly, scalable, and extensible, with the goal of allowing fast experimentation with deep neural networks. It was created as part of the ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) research project, and François Chollet, a Google engineer, is the primary author and maintainer. Chollet is also the creator of the deep neural network model Xception.

4.2.4 DEEP LEARNING MODEL

CONVOLUTIONAL NEURAL NETWORK

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Here we introduced a convolutional neural network for music tagging. Convolutional neural networks are composed of multiple layers of artificial neurons. The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

CNN is patterned to process multidimensional array data in which the convolutional layer takes a stack of feature maps, like the pixels of those color channels, and convolves each feature map with a set of learnable filters to obtain a new stack of output feature maps as input.

Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a “class.”

In a CNN, the input is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) x (feature map height) x (feature map width) x (feature map channels).

Convolutional layer within a CNN generally has the following attributes:

- Convolutional filters/kernels defined by a width and height (hyper-parameters).
- The number of input channels and output channels (hyper-parameters). One layer's input channels must equal the number of output channels (also called depth) of its input.
- Additional hyperparameters of the convolution operation, such as: padding, stride, and dilation.

POOLING LAYER:

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data by reducing the dimensions.

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation.

Accepts a volume of size $w1 \times H1 \times D1$

Requires two hyper parameters:

-their spatial extent F

-the stride S

Produces a volume of size $w2 \times H2 \times D2$

where: $W2 = (W1 - F) / S + 1$

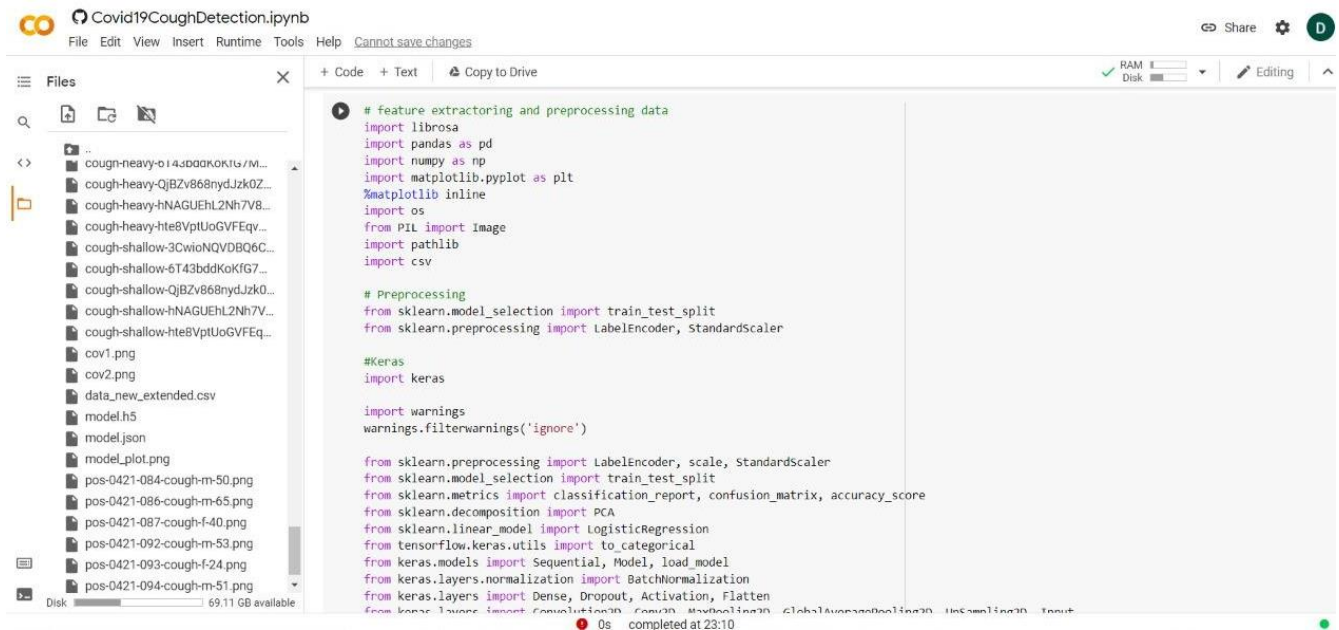
$H2 = (H1 - F) / S + 1$

$D2 = D1$

FULLY CONNECTED LAYER:

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

4.3 MODEL IMPLEMENTATION



The screenshot shows a Jupyter Notebook titled "Covid19CoughDetection.ipynb". The left sidebar displays a file explorer with various files, including audio files (e.g., cough-heavy-0143d00KOKtG/M..., cough-heavy-QBZv868nydJzk0Z..., cough-heavy-hNAGUEHL2Nh7V8..., cough-heavy-hTe8VptUoGVFEq..., cough-shallow-3CwioNQVDBQ6C..., cough-shallow-6T43bddKoKfG7..., cough-shallow-QBZv868nydJzk0..., cough-shallow-hNAGUEHL2Nh7V..., cough-shallow-hTe8VptUoGVFEq..., cov1.png, cov2.png, data_new_extended.csv, model.h5, model.json, model_plot.png, pos-0421-084-cough-m-50.png, pos-0421-086-cough-m-65.png, pos-0421-087-cough-f-40.png, pos-0421-092-cough-m-53.png, pos-0421-093-cough-f-24.png, pos-0421-094-cough-m-51.png) and a disk usage bar indicating 69.11 GB available. The main code area shows the following Python code:

```
# feature extracting and preprocessing data
import librosa
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import os
from PIL import image
import pathlib
import csv

# Preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

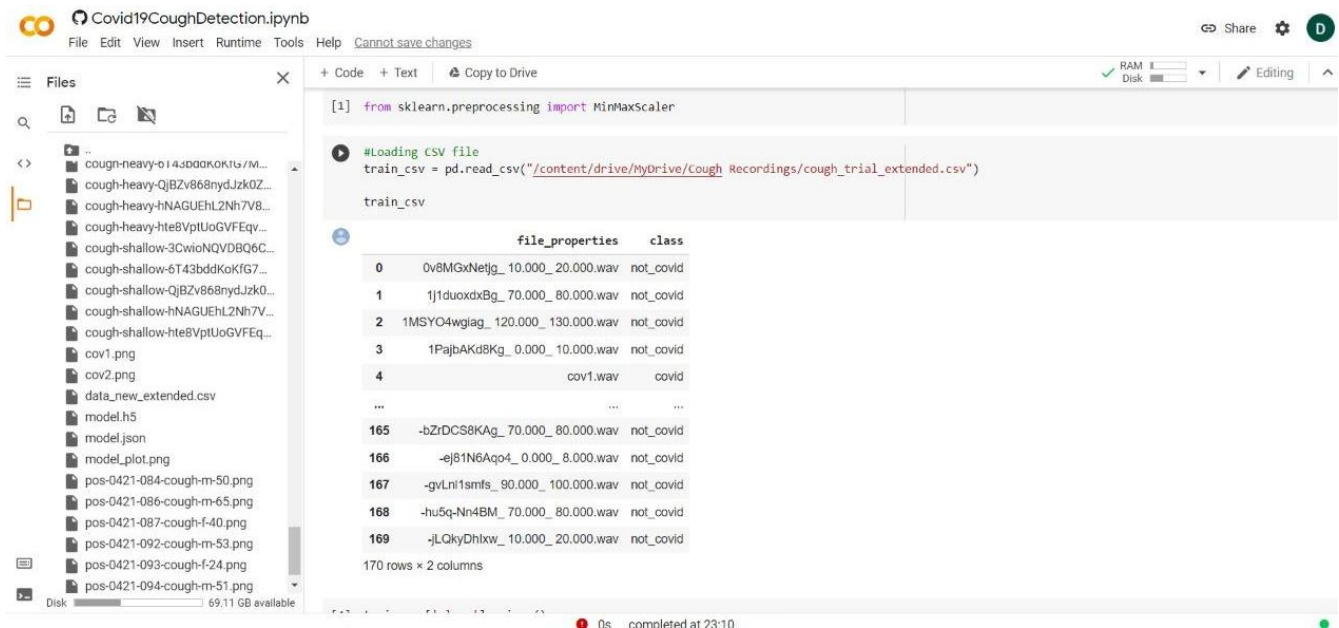
#Keras
import keras

import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import LabelEncoder, scale, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential, Model, load_model
from keras.layers.normalization import BatchNormalization
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution1D, Conv2D, Conv3D, MaxPooling1D, MaxPooling2D, MaxPooling3D, Flatten
```

The code is executed, and the status bar at the bottom indicates "0s completed at 23:10".

Figure 4.1: Importing the required libraries



The screenshot shows the same Jupyter Notebook titled "Covid19CoughDetection.ipynb". The left sidebar is identical to the previous screenshot. The main code area shows the following Python code:

```
[1] from sklearn.preprocessing import MinMaxScaler

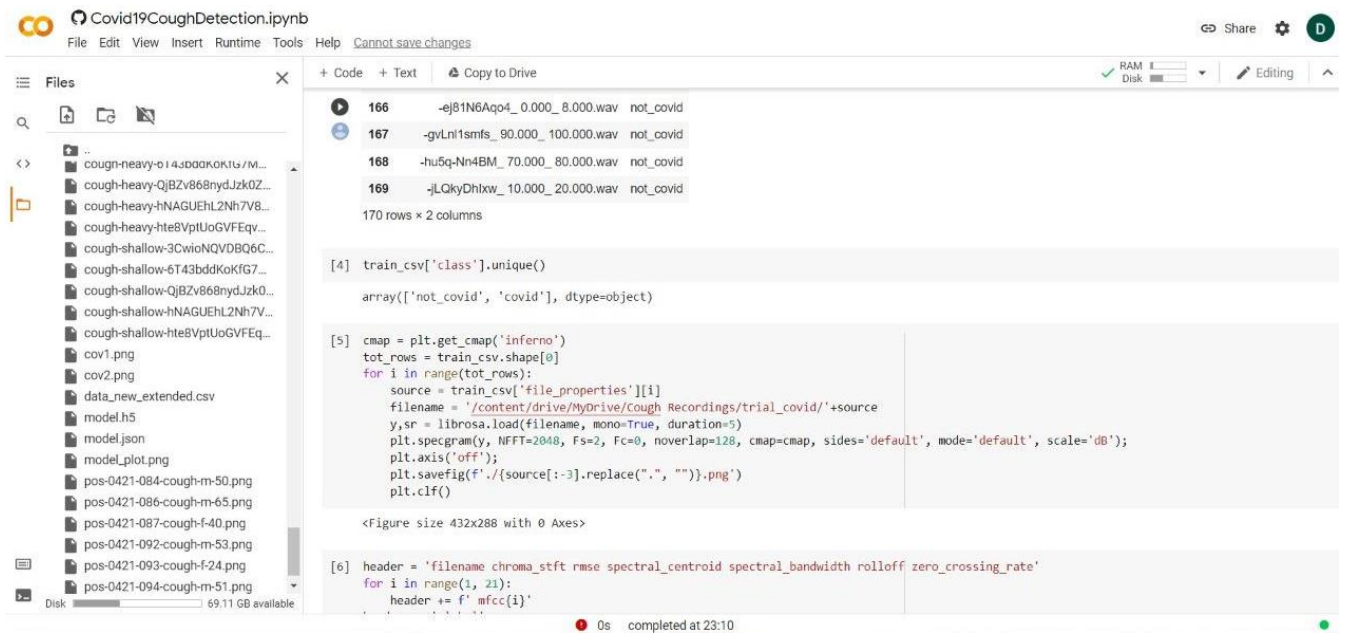
#Loading CSV file
train_csv = pd.read_csv("/content/drive/MyDrive/Cough Recordings/cough_trial_extended.csv")
train_csv
```

The code is executed, and the status bar at the bottom indicates "0s completed at 23:10". The output of the code is a table with 170 rows and 2 columns:

	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
4		cov1.wav
...
165	-bZrDCS8KAg_70.000_80.000.wav	not_covid
166	-ej81N6Aqo4_0.000_8.000.wav	not_covid
167	-gvLn1smfs_90.000_100.000.wav	not_covid
168	-hu5q-Nn4BM_70.000_80.000.wav	not_covid
169	-jLQKyDhIwx_10.000_20.000.wav	not_covid

The status bar at the bottom indicates "0s completed at 23:10".

Figure 4.2: Loading the Sample of the Covid_trial_extended.csv



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with various files, including 'data_new_extended.csv'. The code editor contains the following code:

```
166 -ej81N6Aqo4_0.000_8.000.wav not_covid
167 -gvLn1smfs_90.000_100.000.wav not_covid
168 -hu5q-Nn4BM_70.000_80.000.wav not_covid
169 -jLQkyDhbxw_10.000_20.000.wav not_covid
170 rows x 2 columns

[4] train_csv['class'].unique()

array(['not_covid', 'covid'], dtype=object)

[5] cmap = plt.get_cmap('inferno')
tot_rows = train_csv.shape[0]
for i in range(tot_rows):
    source = train_csv['file_properties'][i]
    filename = '/content/drive/MyDrive/Cough Recordings/trial_covid/' + source
    y, sr = librosa.load(filename, mono=True, duration=5)
    plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, cmap=cmap, sides='default', mode='default', scale='dB');
    plt.axis('off');
    plt.savefig(f'./{source[:-3].replace(".", "")}.png')
    plt.clf()

<Figure size 432x288 with 0 Axes>

[6] header = 'filename chroma_stft rmse spectral_centroid spectral_bandwidth rolloff zero_crossing_rate'
for i in range(1, 21):
    header += f' mfcc{i}'
```

Figure 4.3: Finding the unique classes of train_csv



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with various files, including 'data_new_extended.csv'. The code editor contains the following code:

```
[24] data=pd.read_csv('/content/data_new_extended.csv')

[25] data.shape

(170, 28)

[26] # Dropping unnecesary columns
data = data.drop(['filename'],axis=1)

[27] genre_list = data.iloc[:, -1]
encoder = LabelEncoder()
y = encoder.fit_transform(genre_list)

[28] scaler = StandardScaler()
X = scaler.fit_transform(np.array(data.iloc[:, :-1], dtype = float))

[29] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)

X_train.shape

(113, 26)

[31] y_train.shape

(113,)
```

Figure 4.4: Splitting the data set into train and test set

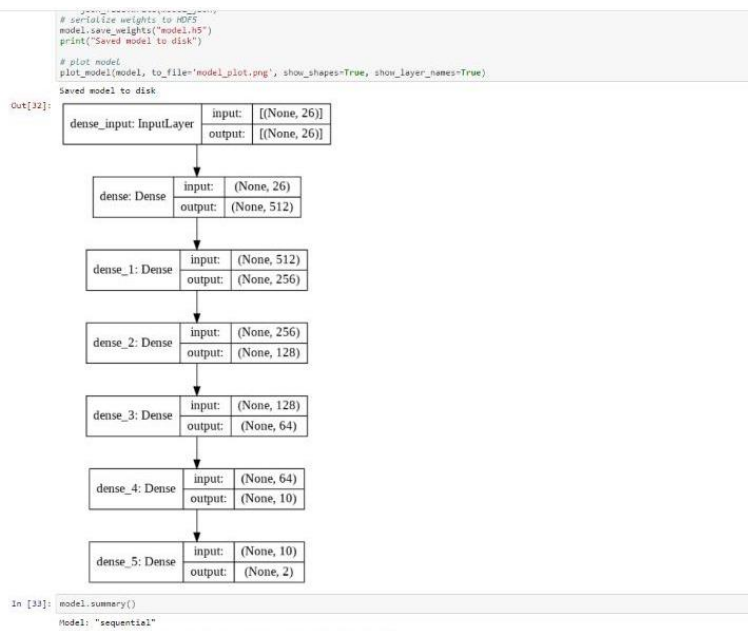


Figure 4.5: Plotting the model

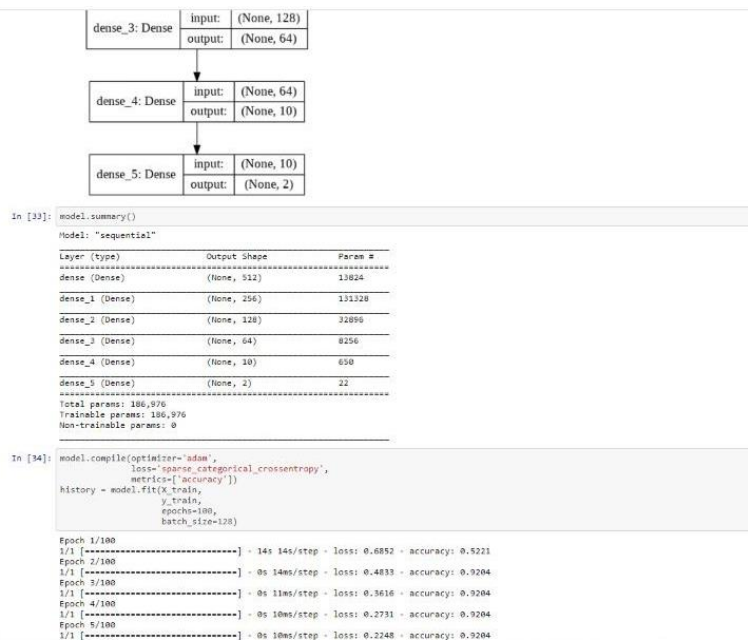


Figure 4.6: Summary of the model

```

Non-trainable params: 0

In [34]: model.compile(optimizer='adam',
                      loss='sparse_categorical_crossentropy',
                      metrics=['accuracy'])
history = model.fit(X_train,
                    y_train,
                    epochs=100,
                    batch_size=128)

Epoch 1/100
1/1 [=====] - 14s 14s/step - loss: 0.6852 - accuracy: 0.5221
Epoch 2/100
1/1 [=====] - 0s 14ms/step - loss: 0.4833 - accuracy: 0.9204
Epoch 3/100
1/1 [=====] - 0s 11ms/step - loss: 0.3616 - accuracy: 0.9204
Epoch 4/100
1/1 [=====] - 0s 10ms/step - loss: 0.2731 - accuracy: 0.9204
Epoch 5/100
1/1 [=====] - 0s 10ms/step - loss: 0.2248 - accuracy: 0.9204
Epoch 6/100
1/1 [=====] - 0s 19ms/step - loss: 0.2043 - accuracy: 0.9204
Epoch 7/100
1/1 [=====] - 0s 10ms/step - loss: 0.1911 - accuracy: 0.9204
Epoch 8/100
1/1 [=====] - 0s 9ms/step - loss: 0.1750 - accuracy: 0.9204
Epoch 9/100
1/1 [=====] - 0s 9ms/step - loss: 0.1543 - accuracy: 0.9204
Epoch 10/100
1/1 [=====] - 0s 9ms/step - loss: 0.1327 - accuracy: 0.9204
Epoch 11/100
1/1 [=====] - 0s 10ms/step - loss: 0.1142 - accuracy: 0.9204
Epoch 12/100
1/1 [=====] - 0s 16ms/step - loss: 0.0999 - accuracy: 0.9204
Epoch 13/100
1/1 [=====] - 0s 20ms/step - loss: 0.0895 - accuracy: 0.9204
Epoch 14/100
1/1 [=====] - 0s 16ms/step - loss: 0.0823 - accuracy: 0.9204
Epoch 15/100
1/1 [=====] - 0s 16ms/step - loss: 0.0777 - accuracy: 0.9204
Epoch 16/100
1/1 [=====] - 0s 17ms/step - loss: 0.0754 - accuracy: 0.9204
Epoch 17/100
1/1 [=====] - 0s 13ms/step - loss: 0.0727 - accuracy: 0.9204
Epoch 18/100
1/1 [=====] - 0s 12ms/step - loss: 0.0692 - accuracy: 0.9204
Epoch 19/100
1/1 [=====] - 0s 13ms/step - loss: 0.0656 - accuracy: 0.9292
Epoch 20/100
1/1 [=====] - 0s 17ms/step - loss: 0.0623 - accuracy: 0.9292
Epoch 21/100
1/1 [=====] - 0s 8ms/step - loss: 0.0596 - accuracy: 0.9381
Epoch 22/100
1/1 [=====] - 0s 11ms/step - loss: 0.0572 - accuracy: 0.9446
Epoch 23/100
1/1 [=====] - 0s 18ms/step - loss: 0.0550 - accuracy: 1.0000
Epoch 24/100
1/1 [=====] - 0s 19ms/step - loss: 0.0530 - accuracy: 1.0000
Epoch 25/100

```

Figure 4.7: Training the model for 100 epochs

```

Epoch 76/100
1/1 [=====] - 0s 13ms/step - loss: 5.1020e-05 - accuracy: 1.0000
Epoch 77/100
1/1 [=====] - 0s 16ms/step - loss: 4.8390e-05 - accuracy: 1.0000
Epoch 78/100
1/1 [=====] - 0s 12ms/step - loss: 4.5992e-05 - accuracy: 1.0000
Epoch 79/100
1/1 [=====] - 0s 10ms/step - loss: 4.3803e-05 - accuracy: 1.0000
Epoch 80/100
1/1 [=====] - 0s 10ms/step - loss: 4.1799e-05 - accuracy: 1.0000
Epoch 81/100
1/1 [=====] - 0s 12ms/step - loss: 3.9963e-05 - accuracy: 1.0000
Epoch 82/100
1/1 [=====] - 0s 10ms/step - loss: 3.8278e-05 - accuracy: 1.0000
Epoch 83/100
1/1 [=====] - 0s 11ms/step - loss: 3.6721e-05 - accuracy: 1.0000
Epoch 84/100
1/1 [=====] - 0s 22ms/step - loss: 3.5195e-05 - accuracy: 1.0000
Epoch 85/100
1/1 [=====] - 0s 13ms/step - loss: 3.3974e-05 - accuracy: 1.0000
Epoch 86/100
1/1 [=====] - 0s 18ms/step - loss: 3.2750e-05 - accuracy: 1.0000
Epoch 87/100
1/1 [=====] - 0s 15ms/step - loss: 3.1626e-05 - accuracy: 1.0000
Epoch 88/100
1/1 [=====] - 0s 13ms/step - loss: 3.0582e-05 - accuracy: 1.0000
Epoch 89/100
1/1 [=====] - 0s 13ms/step - loss: 2.9610e-05 - accuracy: 1.0000
Epoch 90/100
1/1 [=====] - 0s 14ms/step - loss: 2.8697e-05 - accuracy: 1.0000
Epoch 91/100
1/1 [=====] - 0s 19ms/step - loss: 2.7659e-05 - accuracy: 1.0000
Epoch 92/100
1/1 [=====] - 0s 17ms/step - loss: 2.7072e-05 - accuracy: 1.0000
Epoch 93/100
1/1 [=====] - 0s 15ms/step - loss: 2.6337e-05 - accuracy: 1.0000
Epoch 94/100
1/1 [=====] - 0s 14ms/step - loss: 2.5652e-05 - accuracy: 1.0000
Epoch 95/100
1/1 [=====] - 0s 21ms/step - loss: 2.5011e-05 - accuracy: 1.0000
Epoch 96/100
1/1 [=====] - 0s 16ms/step - loss: 2.4410e-05 - accuracy: 1.0000
Epoch 97/100
1/1 [=====] - 0s 13ms/step - loss: 2.3842e-05 - accuracy: 1.0000
Epoch 98/100
1/1 [=====] - 0s 14ms/step - loss: 2.3304e-05 - accuracy: 1.0000
Epoch 99/100
1/1 [=====] - 0s 14ms/step - loss: 2.2800e-05 - accuracy: 1.0000
Epoch 100/100
1/1 [=====] - 0s 16ms/step - loss: 2.2322e-05 - accuracy: 1.0000

In [35]: test_loss, test_acc = model.evaluate(X_test, y_test)
2/2 [=====] - 1s 6ms/step - loss: 0.3467 - accuracy: 0.9298

Test Accuracy

In [36]: print('test_acc: ', test_acc)
test_acc: 0.9298245004870834

```

```

epoch 100/100
1/1 [=====] - 0s 12ms/step - loss: 1.3365e-05 - accuracy: 1.0000

[21]: test_loss, test_acc = model.evaluate(X_test,y_test)
2/2 [=====] - 1s 9ms/step - loss: 0.1328 - accuracy: 0.9825

Test Accuracy

[22]: print('test_acc: ',test_acc)
test_acc: 0.9824561476707458

Predictions on Test Data

[23]: predictions = model.predict(X_test)
print(predictions[0].shape)
print(np.sum(predictions[0]))
print(predictions[:4])
print(y_test[:4])

(2,)
1.0
[[9.9998581e-01 1.4205907e-05]
 [1.5006798e-11 1.0000000e+00]
 [8.3833122e-16 1.0000000e+00]
 [5.4060713e-08 1.0000000e+00]]
[0 1 1 1]

```

Figure 4.8: Making predictions on the test data set

4.4. SUMMARY

This chapter brought out the analysis of each technology used to develop this project. This chapter also brought out the basic system implementations such as the platforms, languages, and tools used here. And the screenshots of different modules implemented using those platforms, languages, and tools.

CHAPTER 5

SYSTEM TESTING AND PERFORMANCE ANALYSIS

5.1 GENERAL

Once the design aspect of the system is finalized the system enters the testing phase. Testing is an integral part of the entire development and maintenance process. Testing is mainly performed to identify errors. It is used for quality assurance. The goal of the testing phase is to verify that the specification has been accurately completed and incorporated into the design, as well as to ensure the correctness of the design itself. Testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and to verify that the software product is fit for use.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- Meets the requirements that guided its design and development
- Responds correctly to all kinds of inputs
- Performs its functions within an acceptable time
- Is sufficiently usable
- Can run in its intended environments
- Achieves the general result of its stakeholder's desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically attempts to execute a program or application with the intent of finding software bugs. The job of testing is an iterative process as when one bug is fixed,

it can illuminate other, deeper bugs, or can even create new ones.

5.2 TEST CASE

Some of the cases are listed below,

- The model is trained to output either 0 or 1 indicating Covid positive or COVID negative
- The input of the model should be the recording of the cough in .wav format

5.3 PERFORMANCE MEASURES

Performance measurement is the process of collecting, analyzing and/or reporting information regarding the performance of an individual, group, organization, system or component. Three metrics are taken into consideration, namely

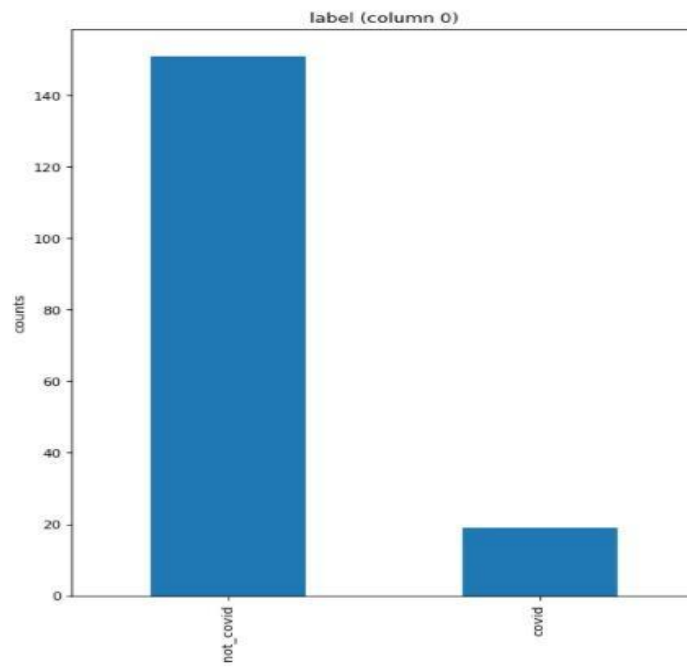
- Training accuracy and loss
- Validation accuracy and loss
- Accuracy score
- F1 score

5.4 PERFORMANCE ANALYSIS

The performance analysis for convolutional neural networks for diagnosis of COVID-19 using cough recordings are analyzed on the train set and test set. The results are as follows.

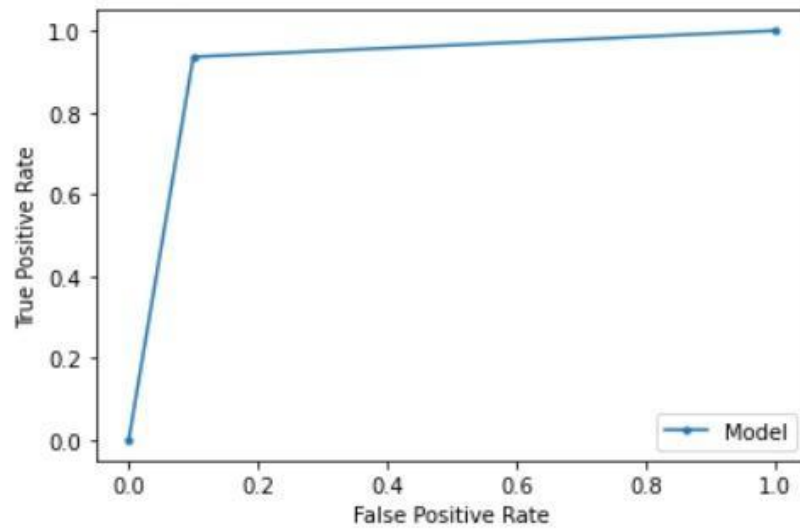
The training accuracy of the model is 0.98% with 0.03% loss. The validation set accuracy of the model is found to be 0.98%. The performance of the model on test set is shown below

Accuracy score: 0.98%

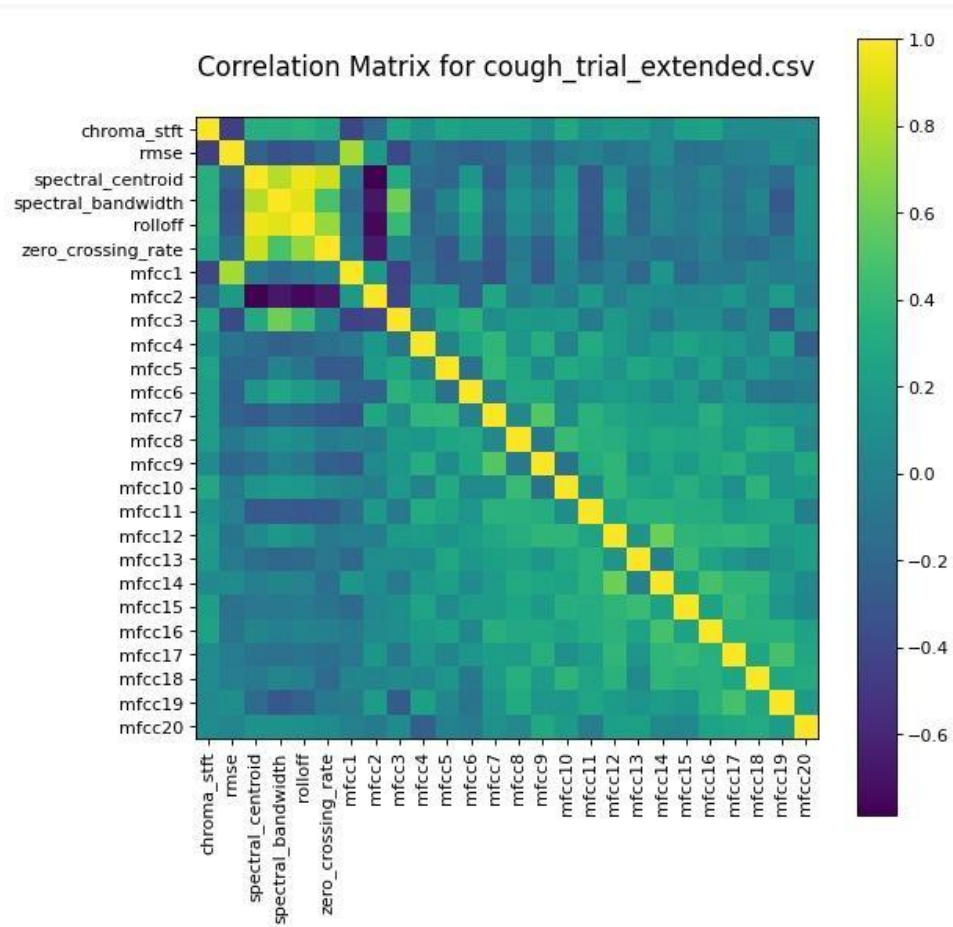


5.4.1 Distribution graphs (histogram/bar graph) of sampled columns

Model: auROC=0.918



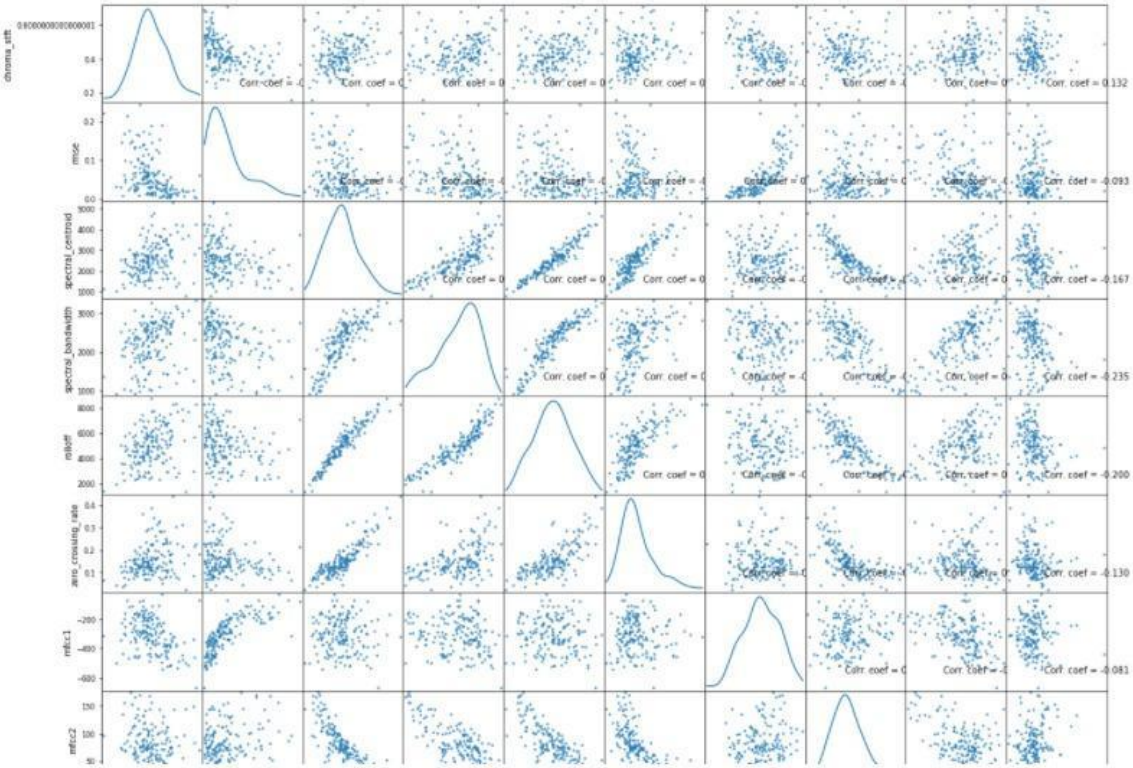
5.4.2 Plotting the roc curve for the model



5.4.3 Correlation Matrix

	Recall	Precision	F1-Score
0	0.9	0.75	0.81818182
1	0.93617021	0.97777778	0.95652174

Table 5.1: Classification Report



5.4.4 Scatter and Density Plot

5.5 SUMMARY

This chapter brought out the general description of the different testing processes applicable to the entire project development. It also considers the performance analysis of a system that proved to improve the level of user satisfaction compared to the existing system.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The system performed really well on detecting COVID-19 and localizing it with 98 percent accuracy. This system helps people to detect and diagnose the disease accurately within seconds. Artificial intelligence is revolutionizing the health care sector all over the world. The AI healthcare market is expected to reach \$45.2 billion USD by 2026 from a current valuation of \$4.9 billion USD. This project plays a vital role in the AI revolution in India, hence reducing the rate of mortality and improving the overall standards of the nation.

6.2 FUTURE WORK

There are many more opportunities for further research in this area, particularly training a classifier with a larger COVID-19 dataset. On further enhancement of the system, we aim to implement a centralized web application for our model which makes the testing process easier and even more accessible to everyone. We'll try looking for more data and better data augmentation techniques, as well as further improving the model.

6.3 APPENDICES

Covid19CoughDetection.ipynb

```
# feature extracting and preprocessing data
import librosa
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import os
```

```

from PIL import Image
import pathlib
import csv

# Preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

#Keras
import keras

import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import LabelEncoder, scale, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential, Model, load_model
from keras.layers.normalization import BatchNormalization
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, Conv2D, MaxPooling2D, GlobalAveragePooling2D,
UpSampling2D, Input
from keras.optimizers import Adam
from keras.utils import np_utils
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from keras import optimizers
from keras.regularizers import l1
from keras.utils.vis_utils import plot_model

```

```

from datetime import datetime
from sklearn import metrics
import librosa, librosa.display, os, csv
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import pylab
plt.switch_backend('agg')
import itertools
import scipy as sp
from scipy import signal
import xgboost as xgb
import joblib
from glob import glob
import urllib

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D, Dropout
from tensorflow.keras.utils import to_categorical


from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC


# Basic Libraries

import pandas as pd
import numpy as np

pd.plotting.register_matplotlib_converters()
import matplotlib.pyplot as plt
%matplotlib inline

```

```

import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV

from sklearn.preprocessing import MinMaxScaler

#Loading CSV file
train_csv = pd.read_csv("/content/drive/MyDrive/Cough Recordings/cough_trial_extended.csv")

train_csv
train_csv['class'].unique()
cmap = plt.get_cmap('inferno')
tot_rows = train_csv.shape[0]
for i in range(tot_rows):
    source = train_csv['file_properties'][i]
    filename = '/content/drive/MyDrive/Cough Recordings/trial_covid/'+source
    y,sr = librosa.load(filename, mono=True, duration=5)
    plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, cmap=cmap, sides='default', mode='default',
scale='dB');
    plt.axis('off');
    plt.savefig(f'./{source[:-3].replace(".", "")}.png')
    plt.clf()
header = 'filename chroma_stft rmse spectral_centroid spectral_bandwidth rolloff zero_crossing_rate'
for i in range(1, 21):
    header += f' mfcc{i}'
header += ' label'
header = header.split()
file = open('/content/drive/MyDrive/Cough Recordings/data_new_extended.csv', 'w')
with file:
    writer = csv.writer(file)

```

```

writer.writerow(header)
for i in range(tot_rows):
    source = train_csv['file_properties'][i]
    file_name = '/content/drive/MyDrive/Cough Recordings/trial_covid/'+source
    y,sr = librosa.load(file_name, 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)
    spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
    rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
    zcr = librosa.feature.zero_crossing_rate(y)
    mfcc = librosa.feature.mfcc(y=y, sr=sr)
    to_append = f'{source[:-3].replace(".", "")} {np.mean(chroma_stft)} {np.mean(rmse)} {np.mean(spec_cent)} {np.mean(spec_bw)} {np.mean(rolloff)} {np.mean(zcr)}'
    for e in mfcc:
        to_append += f' {np.mean(e)}'

    file = open('/content/data_new_extended.csv', 'a')
    with file:
        writer = csv.writer(file)
        writer.writerow(to_append.split())
data=pd.read_csv('/content/data_new_extended.csv')
data.shape
# Dropping unnecessary columns
data = data.drop(['filename'],axis=1)
genre_list = data.iloc[:, -1]
encoder = LabelEncoder()
y = encoder.fit_transform(genre_list)
scaler = StandardScaler()
X = scaler.fit_transform(np.array(data.iloc[:, :-1], dtype = float))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)

```

```

X_train.shape
(113, 26)
y_train.shape
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(512, activation='relu', input_shape=(X_train.shape[1],)))

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

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

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

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

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

# serialize model to JSON
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model.h5")
print("Saved model to disk")

# plot model
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',

```

```

        metrics=['accuracy'])
history = model.fit(X_train,
                    y_train,
                    epochs=100,
                    batch_size=128)
test_loss, test_acc = model.evaluate(X_test,y_test)
print('test_acc: ',test_acc)
predictions = model.predict(X_test)
print(predictions[0].shape)
print(np.sum(predictions[0]))
print(predictions[:4])
print(y_test[:4])
df = pd.DataFrame(predictions, columns = ['Negative','Positive'])
a = df.iloc[:,0] < df.iloc[:,1]
a = a*1
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
a.to_numpy()
from sklearn.metrics import recall_score, precision_score, accuracy_score
from sklearn.metrics import confusion_matrix, f1_score, classification_report
#Evaluation
# Recall - the ability of the classifier to find all the positive samples
print("Recall: ", recall_score(y_test, a,average=None))
# Precision - The precision is intuitively the ability of the classifier not to
#label as positive a sample that is negative
print("Precision: ", precision_score(y_test,a,average=None))

# F1-Score - The F1 score can be interpreted as a weighted average of the precision
#and recall
print("F1-Score: ", f1_score(y_test,a, average=None))

```

```

# Accuracy - the number of correctly classified samples
print("Accuracy: %.2f    ," % accuracy_score(y_test, a,normalize=True), accuracy_score(y_test,
a,normalize=False) )

print("Number of samples:",y_test.shape[0])

# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, a)
np.set_printoptions(precision=2)

#Function to Plot Confusion Matrix
# http://scikit-learn.org/stable/auto\_examples/model\_selection/plot\_confusion\_matrix.html
def plot_confusion_matrix(cm, classes,
                           normalize=False,
                           title='Confusion matrix',
                           cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    """
    #print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)

```



```

plt.yticks(tick_marks, classes)

fmt = '.2f' if normalize else 'd'
thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, format(cm[i, j], fmt),
             horizontalalignment="center",
             color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')

# summarize score
print('Model: auROC=%.3f' % (roc_auc_score(y_test, a)))

# calculate roc curves
lr_fpr, lr_tpr, _ = roc_curve(y_test, a)

# plot the roc curve for the model
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Model')

# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')

# show the legend
pyplot.legend()

# show the plot
pyplot.show()

# Distribution graphs (histogram/bar graph) of column data
def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nunique = df.nunique()
    df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying purposes, pick
    columns that have between 1 and 50 unique values
    nRow, nCol = df.shape

```

```

columnNames = list(df)
nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi = 80, facecolor = 'w',
edgecolor = 'k')
for i in range(min(nCol, nGraphShown)):
    plt.subplot(nGraphRow, nGraphPerRow, i + 1)
    columnDf = df.iloc[:, i]
    if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
        valueCounts = columnDf.value_counts()
        valueCounts.plot.bar()
    else:
        columnDf.hist()
    plt.ylabel('counts')
    plt.xticks(rotation = 90)
    plt.title(f'{columnNames[i]} (column {i})')
plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
plt.show()
# Correlation matrix
def plotCorrelationMatrix(df, graphWidth):
    filename = df.dataframeName
    df = df.dropna('columns') # drop columns with NaN
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique
    values
    if df.shape[1] < 2:
        print(f'No correlation plots shown: The number of non-NaN or constant columns ({df.shape[1]}) is
less than 2')
        return
    corr = df.corr()
    plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor='w', edgecolor='k')
    corrMat = plt.matshow(corr, fignum = 1)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)

```

```

plt.yticks(range(len(corr.columns)), corr.columns)
plt.gca().xaxis.tick_bottom()
plt.colorbar(corrMat)
plt.title(f'Correlation Matrix for {filename}', fontsize=15)
plt.show()

# Scatter and density plots
def plotScatterMatrix(df, plotSize, textSize):
    df = df.select_dtypes(include=[np.number]) # keep only numerical columns
    # Remove rows and columns that would lead to df being singular
    df = df.dropna('columns')
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique
    values
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns for matrix inversion of kernel density plots
        columnNames = columnNames[:10]
    df = df[columnNames]
    ax = pd.plotting.scatter_matrix(df, alpha=0.75, figsize=[plotSize, plotSize], diagonal='kde')
    corrs = df.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction', ha='center',
        va='center', size=textSize)
    plt.suptitle('Scatter and Density Plot')
    plt.show()

nRowsRead = 1000 # specify 'None' if want to read whole file
# cough_trial_extended.csv may have more rows in reality, but we are only loading/previewing the first
1000 rows
df1 = pd.read_csv('/content/data_new_extended.csv', delimiter=',', nrows = nRowsRead)
df1.dataframeName = 'cough_trial_extended.csv'
nRow, nCol = df1.shape
print(f'There are {nRow} rows and {nCol} columns')
plotPerColumnDistribution(df1, 10, 5)

```

```
plotCorrelationMatrix(df1, 8)  
plotScatterMatrix(df1, 20, 10)
```

REFERENCES

- [1] M. B. Alsabek ,A. Hassan, I. and Shahin, "COVID-19 Detection System using Recurrent Neural Networks," 2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), 2020, pp. 1-5, doi: 10.1109/CCCI49893.2020.9256562.
- [2] Guo, J. and Xin, S., "COVID-19 Cough Recognition", Standford University,2020
- [3] Dunne, Rob & Morris,Simon, Tim & Harper. (2020). "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". 10.13140/RG.2.2.13586.89286.
- [4] F. Hueto,J. Laguarda and B. Subirana, "COVID-19 Artificial Intelligence Diagnosis Using Only Cough Recordings," in *IEEE Open Journal of Engineering in Medicine and Biology*, vol. 1, pp. 275-281, 2020, doi: 10.1109/OJEMB.2020.3026928.
- [5]Ali Imran, Charles N. John, Haneya N. Qureshi, Iryna Posokhova, Kamran Ali, MD Iftikhar Hussain, Muhammad Nabeel,Muhammad Sajid Riaz, Usama Masood,AI4COVID-19: AI enabled preliminary diagnosis for COVID-19 from cough samples via an app,Informatics in Medicine Unlocked,Volume 20,2020,100378,ISSN 2352-9148,https://doi.org/10.1016/j.imu.2020.100378.
- [6] <https://www.kaggle.com/himanshu007121/coughclassifier-trial>

