In [3]:

**import** **numpy** **as** **np** *# linear algebra*

**import** **pandas** **as** **pd** *# data processing, CSV file I/O (e.g. pd.read\_csv)*

*# Input data files are available in the read-only "../input/" directory*

*# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory*

**import** **os**

**for** dirname, \_, filenames **in** os.walk('Datasets/Siren\_Audio'):

**for** filename **in** filenames:

print(os.path.join(dirname, filename))

In [4]:

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **os**

**import** **matplotlib.pyplot** **as** **plt**

**import** **tensorflow** **as** **tf**

**from** **tqdm** **import** tqdm *# to see process*

*# Audio Signal Processing Libarary*

**import** **IPython.display** **as** **ipd**

**import** **librosa**

**import** **librosa.display**

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **tensorflow.keras.utils** **import** to\_categorical

**from** **sklearn.preprocessing** **import** LabelEncoder

**from** **tensorflow.keras.models** **import** Sequential

**from** **tensorflow.keras.optimizers** **import** Adam

**from** **tensorflow.keras.layers** **import** Dense, Activation, Conv2D, MaxPool2D, Flatten, Dropout, BatchNormalization

**from** **sklearn** **import** metrics

**from** **sklearn.metrics** **import** classification\_report,confusion\_matrix

In [5]:

path = ['Datasets/fold1','Datasets/fold2','Datasets/fold3']

**for** i **in** range(3):

**for** dirpath, dirname,filename **in** os.walk(path[i]):

print(f"this is **{**i+1**}**st folder having **{**len(filename)**}** sound file in '**{**dirpath**}**'.")

this is 1st folder having 932 sound file in 'Datasets/fold1'.

this is 2st folder having 902 sound file in 'Datasets/fold2'.

this is 3st folder having 429 sound file in 'Datasets/fold3'.

In [6]:

filename = 'Datasets/fold1/ambulance455.wav'

plt.figure(figsize = (14,5))

*## Librosa normalize the sound give it in in one single sample\_rate by deafult this is 22050 or 22KHz*

*#---> and this normalize signal data in 0 to 1 and this change signal into one mono channel.*

*#---> Librosa converts the signal to mono, meaning the channel will alays be 1*

sound\_data, sample\_rate = librosa.load(filename) *# Load file to find data and sr(how many times per sec sound sample)*

print("sample\_rate : ",sample\_rate)

print("data : ",sound\_data)

*# data come in 1-dimensional beacuse librosa change 2 channel into 1 mono channel*

librosa.display.waveshow(sound\_data, sr = sample\_rate) *# Plotting audio file*

plt.title("SINGLE Channel audio signal using LIBROSA")

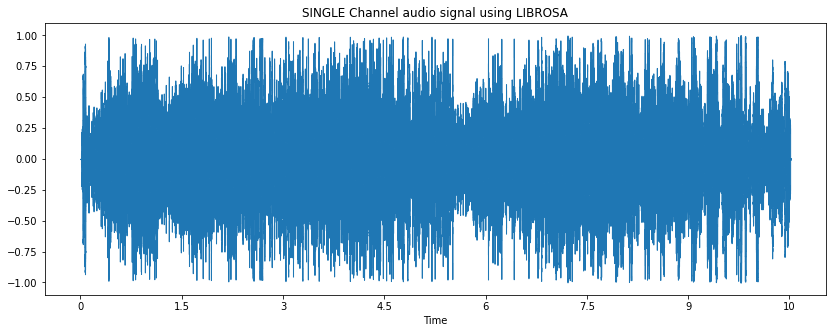
ipd.Audio(filename) *# play the audio*

sample\_rate : 22050

data : [4.4165091e-08 1.8052342e-09 6.5865389e-09 ... 6.6923437e-04 5.7046773e-04

0.0000000e+00]

Out[6]:



In [8]:

data = pd.read\_csv('/Users/akandag/Downloads/Datasets/Dataset.csv')

data.head()

Out[8]:

|  | **slice\_file\_name** | **fold** | **class** |
| --- | --- | --- | --- |
| **0** | ambulance1.wav | 1 | Siren\_Audio |
| **1** | ambulance2.wav | 1 | Siren\_Audio |
| **2** | ambulance3.wav | 1 | Siren\_Audio |
| **3** | ambulance4.wav | 1 | Siren\_Audio |
| **4** | ambulance5.wav | 1 | Siren\_Audio |

In [9]:

data.shape

Out[9]:

(2691, 3)

In [10]:

data['class'].value\_counts()

Out[10]:

Siren\_Audio 932

Road\_Noises\_Audio 901

Car\_Horn\_Audio 858

Name: class, dtype: int64

In [11]:

**def** features\_extract(file\_name):

audio, sample\_rate = librosa.load(file\_name, res\_type = 'kaiser\_fast')

mfccs\_features = librosa.feature.mfcc(y = audio, sr = sample\_rate, n\_mfcc = 40)

mfccs\_scaled\_features = np.mean(mfccs\_features.T,axis = 0)

**return** mfccs\_scaled\_features

In [12]:

pip install resampy

Requirement already satisfied: resampy in /Users/akandag/opt/anaconda3/envs/tensorflow\_env/lib/python3.9/site-packages (0.4.2)

Requirement already satisfied: numba>=0.53 in /Users/akandag/opt/anaconda3/envs/tensorflow\_env/lib/python3.9/site-packages (from resampy) (0.56.4)

Requirement already satisfied: numpy>=1.17 in /Users/akandag/opt/anaconda3/envs/tensorflow\_env/lib/python3.9/site-packages (from resampy) (1.23.5)

Requirement already satisfied: setuptools in /Users/akandag/opt/anaconda3/envs/tensorflow\_env/lib/python3.9/site-packages (from numba>=0.53->resampy) (61.2.0)

Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /Users/akandag/opt/anaconda3/envs/tensorflow\_env/lib/python3.9/site-packages (from numba>=0.53->resampy) (0.39.1)

Note: you may need to restart the kernel to use updated packages.

In [13]:

audio\_dataset\_path = '/Users/akandag/Downloads/Datasets/'

extracted\_features = []

**for** index\_num,row **in** tqdm(data.iterrows()):

file\_name = os.path.join(os.path.abspath(audio\_dataset\_path),'fold'+str(int(row["fold"]))+'/',str(row["slice\_file\_name"]))

final\_class\_labels = row["class"]

data = features\_extract(file\_name)

extracted\_features.append([data,final\_class\_labels])

898it [01:05, 12.79it/s]/Users/akandag/opt/anaconda3/envs/tensorflow\_env/lib/python3.9/site-packages/librosa/core/spectrum.py:256: UserWarning: n\_fft=2048 is too large for input signal of length=1103

warnings.warn(

1919it [02:35, 34.58it/s]/Users/akandag/opt/anaconda3/envs/tensorflow\_env/lib/python3.9/site-packages/librosa/core/spectrum.py:256: UserWarning: n\_fft=2048 is too large for input signal of length=1323

warnings.warn(

2311it [02:48, 57.79it/s]/Users/akandag/opt/anaconda3/envs/tensorflow\_env/lib/python3.9/site-packages/librosa/core/spectrum.py:256: UserWarning: n\_fft=2048 is too large for input signal of length=1523

warnings.warn(

2691it [02:56, 15.28it/s]

In [14]:

features\_df=pd.DataFrame(extracted\_features,columns=['feature','class'])

features\_df.head()

Out[14]:

|  | **feature** | **class** |
| --- | --- | --- |
| **0** | [-410.85336, 19.264332, -50.14613, -40.21634, ... | Siren\_Audio |
| **1** | [-238.44804, 143.10516, -202.82565, -196.61755... | Siren\_Audio |
| **2** | [-355.82266, 102.54215, -81.46596, -14.740185,... | Siren\_Audio |
| **3** | [-382.1787, 97.74871, -52.93776, -19.689226, 4... | Siren\_Audio |
| **4** | [-364.28436, 78.58006, -41.74562, -24.846418, ... | Siren\_Audio |

In [15]:

X = np.array(features\_df['feature'].tolist())

y = np.array(features\_df['class'].tolist())

In [16]:

X.shape, y.shape

Out[16]:

((2691, 40), (2691,))

In [17]:

y

Out[17]:

array(['Siren\_Audio', 'Siren\_Audio', 'Siren\_Audio', ..., 'Car\_Horn\_Audio',

'Car\_Horn\_Audio', 'Car\_Horn\_Audio'], dtype='<U17')

In [18]:

labelencoder = LabelEncoder()

y = to\_categorical(labelencoder.fit\_transform(y)) *# tranform class label*

y

Out[18]:

array([[0., 0., 1.],

[0., 0., 1.],

[0., 0., 1.],

...,

[1., 0., 0.],

[1., 0., 0.],

[1., 0., 0.]], dtype=float32)

In [19]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

print(f'X\_train shape is **{**X\_train.shape**}**')

print(f'X\_test shape is **{**X\_test.shape**}**')

print(f'y\_train shape is **{**y\_train.shape**}**')

print(f'y\_test shape is **{**y\_test.shape**}**')

X\_train shape is (2152, 40)

X\_test shape is (539, 40)

y\_train shape is (2152, 3)

y\_test shape is (539, 3)

In [20]:

labels = y.shape[1] *# total target variable or class variable*

input\_size = X.shape[1] *# total feature value like here n\_mfcc value*

print(f"number of total class label '**{**labels**}**'")

print(f"number of features used '**{**input\_size**}**' ")

number of total class label '3'

number of features used '40'

In [21]:

model = Sequential()

*#first layer*

model.add(Dense(units = 1024, input\_shape = (input\_size,)))

model.add(Activation('relu'))

model.add(Dropout(0.2))

*#second layer*

model.add(Dense(units = 512))

model.add(Activation('relu'))

model.add(Dropout(0.2))

*#third layer*

model.add(Dense(units = 256))

model.add(Activation('relu'))

model.add(Dropout(0.2))

*#final layer*

*# add neural network so flatten the output comming from last layer of cnn model*

model.add(Flatten())

model.add(Dense(units = labels, activation="softmax"))

2023-03-14 03:14:23.244432: I tensorflow/core/platform/cpu\_feature\_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: SSE4.1 SSE4.2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

In [22]:

model.summary()

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense (Dense) (None, 1024) 41984

activation (Activation) (None, 1024) 0

dropout (Dropout) (None, 1024) 0

dense\_1 (Dense) (None, 512) 524800

activation\_1 (Activation) (None, 512) 0

dropout\_1 (Dropout) (None, 512) 0

dense\_2 (Dense) (None, 256) 131328

activation\_2 (Activation) (None, 256) 0

dropout\_2 (Dropout) (None, 256) 0

flatten (Flatten) (None, 256) 0

dense\_3 (Dense) (None, 3) 771

=================================================================

Total params: 698,883

Trainable params: 698,883

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [23]:

model.compile(loss = 'categorical\_crossentropy',

metrics = ['accuracy'],

optimizer = 'adam')

In [24]:

num\_epochs = 150

num\_batch\_size = 32

model.fit(X\_train, y\_train,

batch\_size = num\_batch\_size,

epochs = num\_epochs,

validation\_data = (X\_test, y\_test),

verbose = 1)

Epoch 1/150

68/68 [==============================] - 1s 6ms/step - loss: 2.4137 - accuracy: 0.6468 - val\_loss: 0.5643 - val\_accuracy: 0.8126

Epoch 2/150

68/68 [==============================] - 0s 5ms/step - loss: 0.5828 - accuracy: 0.7923 - val\_loss: 0.3870 - val\_accuracy: 0.8609

Epoch 3/150

68/68 [==============================] - 0s 5ms/step - loss: 0.4585 - accuracy: 0.8234 - val\_loss: 0.3220 - val\_accuracy: 0.8794

Epoch 4/150

68/68 [==============================] - 0s 5ms/step - loss: 0.3641 - accuracy: 0.8601 - val\_loss: 0.2997 - val\_accuracy: 0.9072

Epoch 5/150

68/68 [==============================] - 0s 5ms/step - loss: 0.3234 - accuracy: 0.8792 - val\_loss: 0.3013 - val\_accuracy: 0.9072

Epoch 6/150

68/68 [==============================] - 0s 5ms/step - loss: 0.2829 - accuracy: 0.8973 - val\_loss: 0.2852 - val\_accuracy: 0.8942

Epoch 7/150

68/68 [==============================] - 0s 5ms/step - loss: 0.2469 - accuracy: 0.9080 - val\_loss: 0.2517 - val\_accuracy: 0.9147

Epoch 8/150

68/68 [==============================] - 0s 5ms/step - loss: 0.1966 - accuracy: 0.9294 - val\_loss: 0.1961 - val\_accuracy: 0.9332

Epoch 9/150

68/68 [==============================] - 0s 5ms/step - loss: 0.1843 - accuracy: 0.9387 - val\_loss: 0.1781 - val\_accuracy: 0.9276

Epoch 10/150

68/68 [==============================] - 0s 5ms/step - loss: 0.1600 - accuracy: 0.9391 - val\_loss: 0.1577 - val\_accuracy: 0.9518

Epoch 11/150

68/68 [==============================] - 0s 5ms/step - loss: 0.1652 - accuracy: 0.9414 - val\_loss: 0.1635 - val\_accuracy: 0.9443

Epoch 12/150

68/68 [==============================] - 0s 5ms/step - loss: 0.1380 - accuracy: 0.9540 - val\_loss: 0.1491 - val\_accuracy: 0.9573

Epoch 13/150

68/68 [==============================] - 0s 5ms/step - loss: 0.1317 - accuracy: 0.9531 - val\_loss: 0.1479 - val\_accuracy: 0.9573

Epoch 14/150

68/68 [==============================] - 0s 5ms/step - loss: 0.1223 - accuracy: 0.9600 - val\_loss: 0.1627 - val\_accuracy: 0.9536

Epoch 15/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0945 - accuracy: 0.9647 - val\_loss: 0.1508 - val\_accuracy: 0.9629

Epoch 16/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0912 - accuracy: 0.9661 - val\_loss: 0.1674 - val\_accuracy: 0.9610

Epoch 17/150

68/68 [==============================] - 0s 5ms/step - loss: 0.1060 - accuracy: 0.9582 - val\_loss: 0.1630 - val\_accuracy: 0.9536

Epoch 18/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0955 - accuracy: 0.9675 - val\_loss: 0.1377 - val\_accuracy: 0.9555

Epoch 19/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0747 - accuracy: 0.9744 - val\_loss: 0.1372 - val\_accuracy: 0.9629

Epoch 20/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0562 - accuracy: 0.9782 - val\_loss: 0.1720 - val\_accuracy: 0.9629

Epoch 21/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0661 - accuracy: 0.9772 - val\_loss: 0.2171 - val\_accuracy: 0.9425

Epoch 22/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0936 - accuracy: 0.9670 - val\_loss: 0.1929 - val\_accuracy: 0.9536

Epoch 23/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0786 - accuracy: 0.9698 - val\_loss: 0.1929 - val\_accuracy: 0.9666

Epoch 24/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0487 - accuracy: 0.9837 - val\_loss: 0.2019 - val\_accuracy: 0.9573

Epoch 25/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0745 - accuracy: 0.9730 - val\_loss: 0.1764 - val\_accuracy: 0.9629

Epoch 26/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0352 - accuracy: 0.9884 - val\_loss: 0.1737 - val\_accuracy: 0.9647

Epoch 27/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0421 - accuracy: 0.9870 - val\_loss: 0.1775 - val\_accuracy: 0.9647

Epoch 28/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0454 - accuracy: 0.9828 - val\_loss: 0.2676 - val\_accuracy: 0.9573

Epoch 29/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0649 - accuracy: 0.9814 - val\_loss: 0.1665 - val\_accuracy: 0.9592

Epoch 30/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0570 - accuracy: 0.9796 - val\_loss: 0.1880 - val\_accuracy: 0.9629

Epoch 31/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0304 - accuracy: 0.9875 - val\_loss: 0.2293 - val\_accuracy: 0.9629

Epoch 32/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0482 - accuracy: 0.9861 - val\_loss: 0.2137 - val\_accuracy: 0.9666

Epoch 33/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0516 - accuracy: 0.9847 - val\_loss: 0.1691 - val\_accuracy: 0.9592

Epoch 34/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0542 - accuracy: 0.9837 - val\_loss: 0.2455 - val\_accuracy: 0.9666

Epoch 35/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0233 - accuracy: 0.9907 - val\_loss: 0.2313 - val\_accuracy: 0.9703

Epoch 36/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0350 - accuracy: 0.9879 - val\_loss: 0.2188 - val\_accuracy: 0.9518

Epoch 37/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0461 - accuracy: 0.9833 - val\_loss: 0.2722 - val\_accuracy: 0.9592

Epoch 38/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0345 - accuracy: 0.9888 - val\_loss: 0.1796 - val\_accuracy: 0.9629

Epoch 39/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0275 - accuracy: 0.9884 - val\_loss: 0.2330 - val\_accuracy: 0.9536

Epoch 40/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0479 - accuracy: 0.9819 - val\_loss: 0.1847 - val\_accuracy: 0.9573

Epoch 41/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0467 - accuracy: 0.9819 - val\_loss: 0.2061 - val\_accuracy: 0.9629

Epoch 42/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0323 - accuracy: 0.9902 - val\_loss: 0.2532 - val\_accuracy: 0.9629

Epoch 43/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0296 - accuracy: 0.9907 - val\_loss: 0.1640 - val\_accuracy: 0.9666

Epoch 44/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0228 - accuracy: 0.9926 - val\_loss: 0.2159 - val\_accuracy: 0.9685

Epoch 45/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0223 - accuracy: 0.9949 - val\_loss: 0.2220 - val\_accuracy: 0.9666

Epoch 46/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0170 - accuracy: 0.9944 - val\_loss: 0.1935 - val\_accuracy: 0.9647

Epoch 47/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0155 - accuracy: 0.9954 - val\_loss: 0.2540 - val\_accuracy: 0.9666

Epoch 48/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0404 - accuracy: 0.9875 - val\_loss: 0.1857 - val\_accuracy: 0.9685

Epoch 49/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0350 - accuracy: 0.9865 - val\_loss: 0.2227 - val\_accuracy: 0.9555

Epoch 50/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0344 - accuracy: 0.9893 - val\_loss: 0.2789 - val\_accuracy: 0.9536

Epoch 51/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0516 - accuracy: 0.9814 - val\_loss: 0.2084 - val\_accuracy: 0.9629

Epoch 52/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0481 - accuracy: 0.9842 - val\_loss: 0.2479 - val\_accuracy: 0.9573

Epoch 53/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0278 - accuracy: 0.9902 - val\_loss: 0.2317 - val\_accuracy: 0.9610

Epoch 54/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0421 - accuracy: 0.9898 - val\_loss: 0.1927 - val\_accuracy: 0.9592

Epoch 55/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0277 - accuracy: 0.9888 - val\_loss: 0.2496 - val\_accuracy: 0.9573

Epoch 56/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0569 - accuracy: 0.9819 - val\_loss: 0.1621 - val\_accuracy: 0.9666

Epoch 57/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0463 - accuracy: 0.9865 - val\_loss: 0.1925 - val\_accuracy: 0.9647

Epoch 58/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0388 - accuracy: 0.9861 - val\_loss: 0.2092 - val\_accuracy: 0.9573

Epoch 59/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0254 - accuracy: 0.9916 - val\_loss: 0.2059 - val\_accuracy: 0.9629

Epoch 60/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0162 - accuracy: 0.9930 - val\_loss: 0.1926 - val\_accuracy: 0.9647

Epoch 61/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0077 - accuracy: 0.9958 - val\_loss: 0.2077 - val\_accuracy: 0.9592

Epoch 62/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0123 - accuracy: 0.9963 - val\_loss: 0.2311 - val\_accuracy: 0.9629

Epoch 63/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0082 - accuracy: 0.9967 - val\_loss: 0.2374 - val\_accuracy: 0.9685

Epoch 64/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0146 - accuracy: 0.9963 - val\_loss: 0.2854 - val\_accuracy: 0.9629

Epoch 65/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0386 - accuracy: 0.9875 - val\_loss: 0.2662 - val\_accuracy: 0.9592

Epoch 66/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0279 - accuracy: 0.9930 - val\_loss: 0.3572 - val\_accuracy: 0.9499

Epoch 67/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0719 - accuracy: 0.9777 - val\_loss: 0.2413 - val\_accuracy: 0.9685

Epoch 68/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0506 - accuracy: 0.9828 - val\_loss: 0.2968 - val\_accuracy: 0.9647

Epoch 69/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0350 - accuracy: 0.9902 - val\_loss: 0.2053 - val\_accuracy: 0.9629

Epoch 70/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0323 - accuracy: 0.9888 - val\_loss: 0.2615 - val\_accuracy: 0.9573

Epoch 71/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0833 - accuracy: 0.9786 - val\_loss: 0.2106 - val\_accuracy: 0.9666

Epoch 72/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0118 - accuracy: 0.9954 - val\_loss: 0.3539 - val\_accuracy: 0.9685

Epoch 73/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0266 - accuracy: 0.9921 - val\_loss: 0.2731 - val\_accuracy: 0.9610

Epoch 74/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0159 - accuracy: 0.9954 - val\_loss: 0.2075 - val\_accuracy: 0.9610

Epoch 75/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0315 - accuracy: 0.9930 - val\_loss: 0.2649 - val\_accuracy: 0.9703

Epoch 76/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0313 - accuracy: 0.9902 - val\_loss: 0.3041 - val\_accuracy: 0.9481

Epoch 77/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0194 - accuracy: 0.9935 - val\_loss: 0.2869 - val\_accuracy: 0.9629

Epoch 78/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0387 - accuracy: 0.9902 - val\_loss: 0.1835 - val\_accuracy: 0.9629

Epoch 79/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0291 - accuracy: 0.9902 - val\_loss: 0.2217 - val\_accuracy: 0.9610

Epoch 80/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0211 - accuracy: 0.9916 - val\_loss: 0.3565 - val\_accuracy: 0.9647

Epoch 81/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0107 - accuracy: 0.9967 - val\_loss: 0.3775 - val\_accuracy: 0.9647

Epoch 82/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0025 - accuracy: 1.0000 - val\_loss: 0.3890 - val\_accuracy: 0.9629

Epoch 83/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0032 - accuracy: 0.9986 - val\_loss: 0.4849 - val\_accuracy: 0.9647

Epoch 84/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0104 - accuracy: 0.9967 - val\_loss: 0.3519 - val\_accuracy: 0.9629

Epoch 85/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0199 - accuracy: 0.9940 - val\_loss: 0.3346 - val\_accuracy: 0.9629

Epoch 86/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0111 - accuracy: 0.9958 - val\_loss: 0.4313 - val\_accuracy: 0.9647

Epoch 87/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0110 - accuracy: 0.9972 - val\_loss: 0.4017 - val\_accuracy: 0.9629

Epoch 88/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0128 - accuracy: 0.9963 - val\_loss: 0.3379 - val\_accuracy: 0.9592

Epoch 89/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0409 - accuracy: 0.9893 - val\_loss: 0.3195 - val\_accuracy: 0.9555

Epoch 90/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0291 - accuracy: 0.9907 - val\_loss: 0.3601 - val\_accuracy: 0.9573

Epoch 91/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0292 - accuracy: 0.9907 - val\_loss: 0.3049 - val\_accuracy: 0.9536

Epoch 92/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0307 - accuracy: 0.9902 - val\_loss: 0.3109 - val\_accuracy: 0.9592

Epoch 93/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0226 - accuracy: 0.9940 - val\_loss: 0.3476 - val\_accuracy: 0.9610

Epoch 94/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0203 - accuracy: 0.9944 - val\_loss: 0.2905 - val\_accuracy: 0.9592

Epoch 95/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0395 - accuracy: 0.9916 - val\_loss: 0.1981 - val\_accuracy: 0.9629

Epoch 96/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0090 - accuracy: 0.9967 - val\_loss: 0.3456 - val\_accuracy: 0.9610

Epoch 97/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0080 - accuracy: 0.9972 - val\_loss: 0.3706 - val\_accuracy: 0.9481

Epoch 98/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0129 - accuracy: 0.9967 - val\_loss: 0.3204 - val\_accuracy: 0.9629

Epoch 99/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0068 - accuracy: 0.9977 - val\_loss: 0.3315 - val\_accuracy: 0.9610

Epoch 100/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0256 - accuracy: 0.9930 - val\_loss: 0.3616 - val\_accuracy: 0.9629

Epoch 101/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0220 - accuracy: 0.9949 - val\_loss: 0.2721 - val\_accuracy: 0.9647

Epoch 102/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0098 - accuracy: 0.9981 - val\_loss: 0.3587 - val\_accuracy: 0.9685

Epoch 103/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0362 - accuracy: 0.9902 - val\_loss: 0.3185 - val\_accuracy: 0.9685

Epoch 104/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0279 - accuracy: 0.9926 - val\_loss: 0.3583 - val\_accuracy: 0.9629

Epoch 105/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0365 - accuracy: 0.9902 - val\_loss: 0.4515 - val\_accuracy: 0.9647

Epoch 106/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0443 - accuracy: 0.9893 - val\_loss: 0.2775 - val\_accuracy: 0.9573

Epoch 107/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0211 - accuracy: 0.9926 - val\_loss: 0.2467 - val\_accuracy: 0.9610

Epoch 108/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0193 - accuracy: 0.9972 - val\_loss: 0.2553 - val\_accuracy: 0.9629

Epoch 109/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0389 - accuracy: 0.9907 - val\_loss: 0.2443 - val\_accuracy: 0.9592

Epoch 110/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0181 - accuracy: 0.9958 - val\_loss: 0.2648 - val\_accuracy: 0.9592

Epoch 111/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0431 - accuracy: 0.9893 - val\_loss: 0.2445 - val\_accuracy: 0.9629

Epoch 112/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0209 - accuracy: 0.9912 - val\_loss: 0.2034 - val\_accuracy: 0.9666

Epoch 113/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0080 - accuracy: 0.9972 - val\_loss: 0.3127 - val\_accuracy: 0.9629

Epoch 114/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0255 - accuracy: 0.9921 - val\_loss: 0.2378 - val\_accuracy: 0.9647

Epoch 115/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0099 - accuracy: 0.9972 - val\_loss: 0.3161 - val\_accuracy: 0.9610

Epoch 116/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0343 - accuracy: 0.9926 - val\_loss: 0.3143 - val\_accuracy: 0.9685

Epoch 117/150

68/68 [==============================] - 0s 4ms/step - loss: 0.0061 - accuracy: 0.9977 - val\_loss: 0.3672 - val\_accuracy: 0.9629

Epoch 118/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0183 - accuracy: 0.9958 - val\_loss: 0.4528 - val\_accuracy: 0.9518

Epoch 119/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0366 - accuracy: 0.9888 - val\_loss: 0.3069 - val\_accuracy: 0.9629

Epoch 120/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0342 - accuracy: 0.9916 - val\_loss: 0.4239 - val\_accuracy: 0.9555

Epoch 121/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0476 - accuracy: 0.9893 - val\_loss: 0.2446 - val\_accuracy: 0.9610

Epoch 122/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0107 - accuracy: 0.9963 - val\_loss: 0.2787 - val\_accuracy: 0.9610

Epoch 123/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0101 - accuracy: 0.9977 - val\_loss: 0.2893 - val\_accuracy: 0.9592

Epoch 124/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0028 - accuracy: 0.9991 - val\_loss: 0.3362 - val\_accuracy: 0.9592

Epoch 125/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0052 - accuracy: 0.9986 - val\_loss: 0.3228 - val\_accuracy: 0.9592

Epoch 126/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0038 - accuracy: 0.9981 - val\_loss: 0.2921 - val\_accuracy: 0.9610

Epoch 127/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0065 - accuracy: 0.9977 - val\_loss: 0.4104 - val\_accuracy: 0.9647

Epoch 128/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0267 - accuracy: 0.9916 - val\_loss: 0.3818 - val\_accuracy: 0.9610

Epoch 129/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0279 - accuracy: 0.9898 - val\_loss: 0.2787 - val\_accuracy: 0.9592

Epoch 130/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0062 - accuracy: 0.9981 - val\_loss: 0.3574 - val\_accuracy: 0.9629

Epoch 131/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0045 - accuracy: 0.9981 - val\_loss: 0.3769 - val\_accuracy: 0.9592

Epoch 132/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0049 - accuracy: 0.9981 - val\_loss: 0.4819 - val\_accuracy: 0.9555

Epoch 133/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0292 - accuracy: 0.9902 - val\_loss: 0.3687 - val\_accuracy: 0.9592

Epoch 134/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0217 - accuracy: 0.9930 - val\_loss: 0.3544 - val\_accuracy: 0.9740

Epoch 135/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0215 - accuracy: 0.9926 - val\_loss: 0.3808 - val\_accuracy: 0.9462

Epoch 136/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0274 - accuracy: 0.9916 - val\_loss: 0.2615 - val\_accuracy: 0.9666

Epoch 137/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0247 - accuracy: 0.9912 - val\_loss: 0.3267 - val\_accuracy: 0.9666

Epoch 138/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0127 - accuracy: 0.9954 - val\_loss: 0.2438 - val\_accuracy: 0.9610

Epoch 139/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0181 - accuracy: 0.9916 - val\_loss: 0.2966 - val\_accuracy: 0.9685

Epoch 140/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0203 - accuracy: 0.9944 - val\_loss: 0.2026 - val\_accuracy: 0.9610

Epoch 141/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0033 - accuracy: 0.9981 - val\_loss: 0.2565 - val\_accuracy: 0.9592

Epoch 142/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0075 - accuracy: 0.9972 - val\_loss: 0.3333 - val\_accuracy: 0.9666

Epoch 143/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0120 - accuracy: 0.9963 - val\_loss: 0.2325 - val\_accuracy: 0.9666

Epoch 144/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0200 - accuracy: 0.9944 - val\_loss: 0.2441 - val\_accuracy: 0.9685

Epoch 145/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0074 - accuracy: 0.9977 - val\_loss: 0.2992 - val\_accuracy: 0.9629

Epoch 146/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0070 - accuracy: 0.9967 - val\_loss: 0.5272 - val\_accuracy: 0.9647

Epoch 147/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0224 - accuracy: 0.9930 - val\_loss: 0.3169 - val\_accuracy: 0.9629

Epoch 148/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0175 - accuracy: 0.9944 - val\_loss: 0.2845 - val\_accuracy: 0.9666

Epoch 149/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0155 - accuracy: 0.9944 - val\_loss: 0.3274 - val\_accuracy: 0.9610

Epoch 150/150

68/68 [==============================] - 0s 5ms/step - loss: 0.0356 - accuracy: 0.9916 - val\_loss: 0.3340 - val\_accuracy: 0.9555

Out[24]:

<keras.callbacks.History at 0x7fb28b454910>

In [25]:

loss = pd.DataFrame(model.history.history)

*#plotting the loss and accuracy*

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

plt.subplot(2,2,1)

plt.plot(loss["loss"], label ="Loss")

plt.plot(loss["val\_loss"], label = "Validation\_loss")

plt.legend()

plt.title("Training and Validation Loss")

plt.subplot(2,2,2)

plt.plot(loss['accuracy'],label = "Training Accuracy")

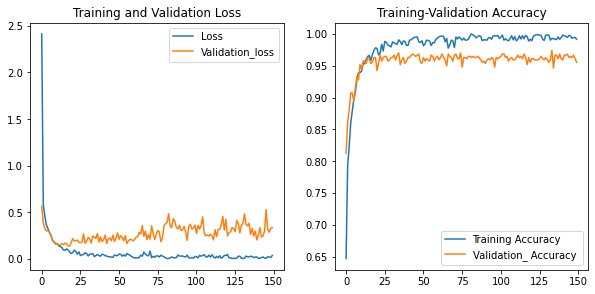
plt.plot(loss['val\_accuracy'], label ="Validation\_ Accuracy ")

plt.legend()

plt.title("Training-Validation Accuracy")

Out[25]:

Text(0.5, 1.0, 'Training-Validation Accuracy')



In [26]:

prediction = model.predict(X\_test)

*# finding class with larget predicted probability using argmax of numpy*

y\_pred = np.argmax(prediction, axis = 1) *# prediction using model*

y\_test\_orig = np.argmax(y\_test, axis = 1) *# original y\_test*

print(y\_pred)

17/17 [==============================] - 0s 2ms/step

[2 2 0 0 2 2 2 2 2 0 2 2 0 2 1 0 0 1 2 1 0 1 0 2 2 2 2 1 2 1 0 0 2 2 1 0 2

2 1 0 1 2 1 1 1 2 1 2 1 2 0 2 0 1 0 0 1 1 0 1 0 1 2 2 1 0 0 1 0 2 0 2 0 1

1 2 1 2 0 2 0 2 0 1 2 0 0 1 2 0 2 0 1 0 0 2 2 2 2 2 2 1 0 1 2 2 0 1 0 1 2

1 1 2 0 0 0 2 2 1 1 1 1 2 2 0 0 1 2 0 1 2 0 2 2 2 1 2 2 2 2 1 1 0 0 1 2 0

2 1 2 0 2 2 1 1 0 2 0 0 0 2 0 2 1 0 1 1 1 2 1 0 2 2 0 1 0 0 2 2 1 1 1 2 0

2 0 0 0 1 0 1 0 2 0 1 2 0 1 2 1 0 1 1 2 0 1 1 1 1 1 1 0 2 0 2 1 2 1 0 0 1

2 1 2 0 2 0 1 0 1 2 2 2 0 2 2 2 0 0 2 1 1 0 0 2 1 2 1 0 1 1 1 2 1 0 0 1 1

1 0 0 1 2 1 1 0 2 0 1 0 2 1 1 0 2 1 1 1 0 0 1 1 2 1 1 0 0 2 1 2 2 0 2 1 0

0 2 1 0 1 0 1 1 2 1 2 2 2 1 1 2 0 0 2 0 2 0 1 0 2 0 0 2 1 1 2 0 2 1 1 1 2

2 0 0 2 2 0 0 2 2 2 1 2 2 0 1 2 0 1 0 0 1 2 2 0 0 2 0 1 2 2 0 1 1 2 0 1 2

1 0 1 2 0 2 0 2 0 2 0 0 1 1 0 0 2 2 0 2 2 2 2 0 1 2 1 1 2 2 1 2 0 2 2 2 1

2 2 2 0 0 0 0 2 1 2 1 2 0 0 0 0 0 2 2 1 0 1 0 2 2 0 2 2 2 0 2 0 0 2 0 1 1

1 1 0 0 1 2 0 1 1 1 2 0 1 1 1 0 2 0 0 1 0 0 0 2 1 1 1 0 0 1 1 1 2 0 1 2 1

1 1 0 0 0 2 1 0 0 1 1 0 2 2 0 0 0 1 2 2 2 1 2 0 1 2 1 2 1 2 2 1 2 2 2 0 0

2 2 2 2 2 2 0 1 2 2 2 0 0 0 2 1 2 1 1 1 0]

In [27]:

class\_label\_lst = np.array(features\_df['class'].unique().tolist())

print(class\_label\_lst)

['Siren\_Audio' 'Road\_Noises\_Audio' 'Car\_Horn\_Audio']

In [28]:

class\_name = ['fold1','fold2','fold3']

print(classification\_report(y\_test\_orig, y\_pred, target\_names = class\_name))

precision recall f1-score support

fold1 0.98 1.00 0.99 171

fold2 0.95 0.92 0.93 177

fold3 0.93 0.95 0.94 191

accuracy 0.96 539

macro avg 0.96 0.96 0.96 539

weighted avg 0.96 0.96 0.96 539

In [29]:

confusion\_df = pd.DataFrame(confusion\_matrix(y\_test\_orig, y\_pred), columns = class\_name, index = class\_name)

print("**\n**")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONFUSION METRIX \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("**\n**")

confusion\_df

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONFUSION METRIX \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Out[29]:

|  | **fold1** | **fold2** | **fold3** |
| --- | --- | --- | --- |
| **fold1** | 171 | 0 | 0 |
| **fold2** | 2 | 162 | 13 |
| **fold3** | 1 | 8 | 182 |

In [30]:

**from** **numpy** **import** loadtxt

**from** **keras.models** **import** load\_model

In [32]:

model.save("model.v5")

print("Saved model to disk")

INFO:tensorflow:Assets written to: model.v5/assets

Saved model to disk

In [33]:

model = load\_model('model.v5')

In [34]:

classes = features\_df.groupby('class')['class'].unique()

classes

Out[34]:

class

Car\_Horn\_Audio [Car\_Horn\_Audio]

Road\_Noises\_Audio [Road\_Noises\_Audio]

Siren\_Audio [Siren\_Audio]

Name: class, dtype: object

In [51]:

**def** predict(path):

audio = np.array([features\_extract(path)])

classid = np.argmax(model.predict(audio)[0])

print('Class predicted :', classes[classid][0],'**\n\n**')

**return** ipd.Audio(path)

In [41]:

!pip install pyserial

**import** **serial**

**import** **time**

Requirement already satisfied: pyserial in /Users/akandag/opt/anaconda3/envs/tensorflow\_env/lib/python3.9/site-packages (3.5)

In [49]:

**while** **True**:

audio = np.array([features\_extract("/Users/akandag/Downloads/output.wav")])

classid = np.argmax(model.predict(audio)[0])

print(classes[classid][0])

ser = serial.Serial('/dev/cu.usbmodem1301', 9600)

**if** (classes[classid][0] == 'Siren\_Audio'):

ser.write(b'1')

**elif** (classes[classid][0] == 'Car\_Horn\_Audio'):

ser.write(b'2')

*# else:*

*# ser.write(b'2')*

time.sleep(3)

1/1 [==============================] - 0s 15ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 11ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 12ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 16ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 15ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 14ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 11ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 17ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 12ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 13ms/step

Car\_Horn\_Audio

1/1 [==============================] - 0s 17ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 13ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 15ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 14ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 12ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 12ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 12ms/step

Siren\_Audio

1/1 [==============================] - 0s 15ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 12ms/step

Siren\_Audio

1/1 [==============================] - 0s 15ms/step

Siren\_Audio

1/1 [==============================] - 0s 12ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 14ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 14ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 16ms/step

Road\_Noises\_Audio

1/1 [==============================] - 0s 14ms/step

Road\_Noises\_Audio

---------------------------------------------------------------------------

KeyboardInterrupt Traceback (most recent call last)

Input In [49], in <cell line: 2>()

9 ser.write(b'2')

10 # else:

11 # ser.write(b'2')

---> 12 time.sleep(3)

KeyboardInterrupt:

In [43]:

**import** **sounddevice** **as** **sd**

**from** **scipy.io.wavfile** **import** write

In [44]:

fs = 16000

seconds = 3

In [50]:

**while** **True**:

audio = np.array([features\_extract("/Users/akandag/Downloads/output.wav")])

classid = np.argmax(model.predict(audio)[0])

print(classes[classid][0])

ser = serial.Serial('/dev/cu.usbmodem1301', 9600)

**if** (classes[classid][0] == 'Siren\_Audio'):

ser.write(b'1')

**elif** (classes[classid][0] == 'Car\_Horn\_Audio'):

ser.write(b'2')

time.sleep(3)