MOSAIC'23 PS-1

CRACKLES AND WHEEZES PREDICTOR

@442|PURAV HARAN|PRASOON SAHAY|PRIYANSHU

OVERVIEW

-> Load data
-> Importing necessary libraries
-> Audio Splitting
-> Making the dataframe
-> Converting audio to numerical features
-> Reshaping the features
-> CONVERTING THE TRAIN DATA
-> SPLITTING THE TRAINING AND TESTING DATA
-> PREPARING THE MODEL
-> FITTING THE MODEL
-> EVALUATING THE MODEL

LOAD DATA

-> We will upload the folder in the google drive and from there we will mount the google drive in folder section of google colab.

-> Then we can easily access it through its path from folder section of google drive.

IMPORTING NECESSARY LIBRARIES

Importing necessary libraries

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
%matplotlib inline

# Don't Show Warning Messages
import warnings
warnings.filterwarnings('ignore')
from datetime import datetime
from os import listdir
from os.path import isfile, join

import librosa
import librosa.display

import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv2D, MaxPooling2D, GlobalAveragePooling2D
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import ModelCheckpoint

from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

import matplotlib.pyplot as plt
import seaborn as sns
```

MAKING A DATAFRAME

Making a dataframe

```
col_names = ['Beginning_of_respiratory_cycle', 'End_of_respiratory_cycle', 'Presence/absence_of_crackles', 'Presence/absence_of_wheezes']
i_list = []
rec_annotations = []
rec_annotations_dict = {}
for s in filenames:
    path = \
    '/content/gdrive/MyDrive/ICBHI_final_database/audio_and_text/'+s+'.txt'
    df_annot = pd.read_csv(path, sep="\t", header=None, names=col_names)
    i = Extract_Annotation_Data(s,root)
    i_list.append(i)
    i_list.append(df_annot)
    # rec_annotations.append(a)
    # rec_annotations_dict[s] = a
recording_info = pd.concat(i_list, axis = 0)
recording_info[['Patient number', 'Recording index', 'Chest location','Acquisition mode','Recording equipment']]=recording_info[['Patient number', 'Recording index'
recording_info=recording_info.dropna()
recording_info= (variable) recording_info: DataFrame | Any
recording_info=recording_info.drop('index',axis=1)
my_index = pd.Series(list(rec_annotations_dict.keys()))
recording_info.head(100)
```

[79]
```
recording_info=recording_info.drop('index',axis=1)
my_index = pd.Series(list(rec_annotations_dict.keys()))
recording_info.head(100)
```

	Patient number	Recording index	Chest location	Acquisition mode	Recording equipment	Beginning_of_respiratory_cycle	End_of_respiratory_cycle	Presence/absence_of_crackles	Presence/absence_of
0	130	3p2	Pr	mc	AKGC417L	1.461	2.918	1.0	
1	130	3p2	Pr	mc	AKGC417L	2.918	5.839	1.0	
2	130	3p2	Pr	mc	AKGC417L	5.839	8.605	1.0	
3	130	3p2	Pr	mc	AKGC417L	8.605	11.637	1.0	
4	130	3p2	Pr	mc	AKGC417L	11.637	14.492	1.0	
...	...	...	...	...	...	...	...	...	
95	130	3p4	Tc	mc	AKGC417L	2.697	5.470	1.0	
96	130	3p4	Tc	mc	AKGC417L	5.470	7.995	1.0	
97	130	3p4	Tc	mc	AKGC417L	7.995	10.729	1.0	
98	130	3p4	Tc	mc	AKGC417L	10.729	13.377	1.0	
99	130	3p4	Tc	mc	AKGC417L	13.377	15.775	1.0	

100 rows × 9 columns

```
recording_info['r']=''
for ind in recording_info.index:
  if(recording_info['Presence/absence_of_crackles'][ind]==0 and recording_info['Presence/absence_of_wheezes'][ind]==0):
    recording_info['r'][ind]=0
  elif(recording_info['Presence/absence_of_crackles'][ind]==0 and recording_info['Presence/absence_of_wheezes'][ind]==1):
    recording_info['r'][ind]=1
  elif(recording_info['Presence/absence_of_crackles'][ind]==1 and recording_info['Presence/absence_of_wheezes'][ind]==0):
    recording_info['r'][ind]=2
  elif(recording_info['Presence/absence_of_crackles'][ind]==1 and recording_info['Presence/absence_of_wheezes'][ind]==1):
    recording_info['r'][ind]=3
recording_info.head(100)
```

	Patient number	Recording index	Chest location	Acquisition mode	Recording equipment	Beginning_of_respiratory_cycle	End_of_respiratory_cycle	Presence/absence_of_crackles	Presence/absence_of_wheezes	r
0	130	3p2	Pr	mc	AKGC417L	1.461	2.918	1.0	0.0	2
1	130	3p2	Pr	mc	AKGC417L	2.918	5.839	1.0	0.0	2
2	130	3p2	Pr	mc	AKGC417L	5.839	8.605	1.0	0.0	2
3	130	3p2	Pr	mc	AKGC417L	8.605	11.637	1.0	0.0	2
4	130	3p2	Pr	mc	AKGC417L	11.637	14.492	1.0	0.0	2
...	...	...	...	...	...	...	...	...	...	...
95	130	3p4	Tc	mc	AKGC417L	2.697	5.470	1.0	1.0	3
96	130	3p4	Tc	mc	AKGC417L	5.470	7.995	1.0	1.0	3
97	130	3p4	Tc	mc	AKGC417L	7.995	10.729	1.0	1.0	3
98	130	3p4	Tc	mc	AKGC417L	10.729	13.377	1.0	1.0	3
99	130	3p4	Tc	mc	AKGC417L	13.377	15.775	1.0	0.0	2

100 rows × 10 columns

-> We are splitting the filenames and inserting them in each columns of a new dataframe so we easily access them by iterating through the dataframe.

-> Then we will encode the presence of crackles and wizzes and make a new column named ‘r’.

AUDIO SPLITTING

AUDIO SPLITTING

```
for ind in recording_info.index:
    audio_file = recording_info['Patient number'][ind]+'_'+recording_info['Recording index'][ind]+'_'+recording_info['Chest location'][ind]+'_'+recording_i
    audio_file1= recording_info['Patient number'][ind]+'_'+recording_info['Recording index'][ind]+'_'+recording_info['Chest location'][ind]+'_'+recording_i

    path = \
    '/content/gdrive/MyDrive/ICBHI_final_database/audio_and_text/' + audio_file

    # read the file
    data, rate = sf.read(path)
    start_time = recording_info['Beginning_of_respiratory_cycle'][ind]
    end_time = recording_info['End_of_respiratory_cycle'][ind]

    t1 = start_time * 1000 # pydub works in milliseconds
    t2 = end_time * 1000
    newAudio = AudioSegment.from_wav(path) # path is defined above
    newAudio = newAudio[t1:t2]
    newAudio.export(audio_file1, format="wav")
```

-> We will split the audio files according to the beginning and ending of the respiratory cycle.

-> And then rename by adding the string beginning of the respiratory cycle to the previous name.

CONVERTING AUDIO TO NUMERICAL FEATURES

Converting audio to numerical features

```
max_pad_len = 862 # to make the length of all MFCC equal
def extract_features(file_name):
    """
    # This function takes in the path for an audio file as a string, loads it, and returns the MFCC
    # of the audio"""

    try:
        audio, sample_rate = librosa.load(file_name,duration=20)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
        pad_width = max_pad_len - mfccs.shape[1]
        mfccs = np.pad(mfccs, pad_width=((0, 0), (0, pad_width)), mode='constant')

    except Exception as e:
        print("Error encountered while parsing file: ", file_name)
        return None
    # audio, sample_rate = librosa.load(file_name)
    # 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
    return mfccs
```

```
[85] features = []
     import os
     # Iterate through each sound file and extract the features
     for ind in recording_info.index:
         audio_file1= recording_info['Patient number'][ind]+'_'+recording_info['Recording index'][ind]+'_'+recording_info['Chest location'][ind]+'_'+recording_info['Acquisition mode'][ind]+'_'+recording_info['Recording equipme

         path = \
         '/content/' + audio_file1
         data = extract_features(path)
         features.append(data)
     print('Finished feature extraction from ', len(features), ' files')
     features = np.array(features)
```

PREPARING THE MODEL

PREPARING THE MODEL

```
num_rows = 40
num_columns = 862
num_channels = 1

num_labels = labels.shape[1]
filter_size = 2
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=filter_size,
                 input_shape=(num_rows, num_columns, num_channels), activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv2D(filters=32, kernel_size=filter_size, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv2D(filters=64, kernel_size=filter_size, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv2D(filters=128, kernel_size=filter_size, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))
model.add(GlobalAveragePooling2D())

model.add(Dense(num_labels, activation='softmax'))
```

```
[103] model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
```

We will implement the CNN model to train the data.

FITTING THE MODEL

FITTING THE MODEL

```
nepochs = 70
num_batch_size = 128
callbacks = [
    ModelCheckpoint(
        filepath='./model./model1.h5',
        save_best_only=True,
        monitor='val_accuracy',
        verbose=1)
]
model.fit(X_train, Y_train,epochs=nepochs,callbacks=callbacks)
```

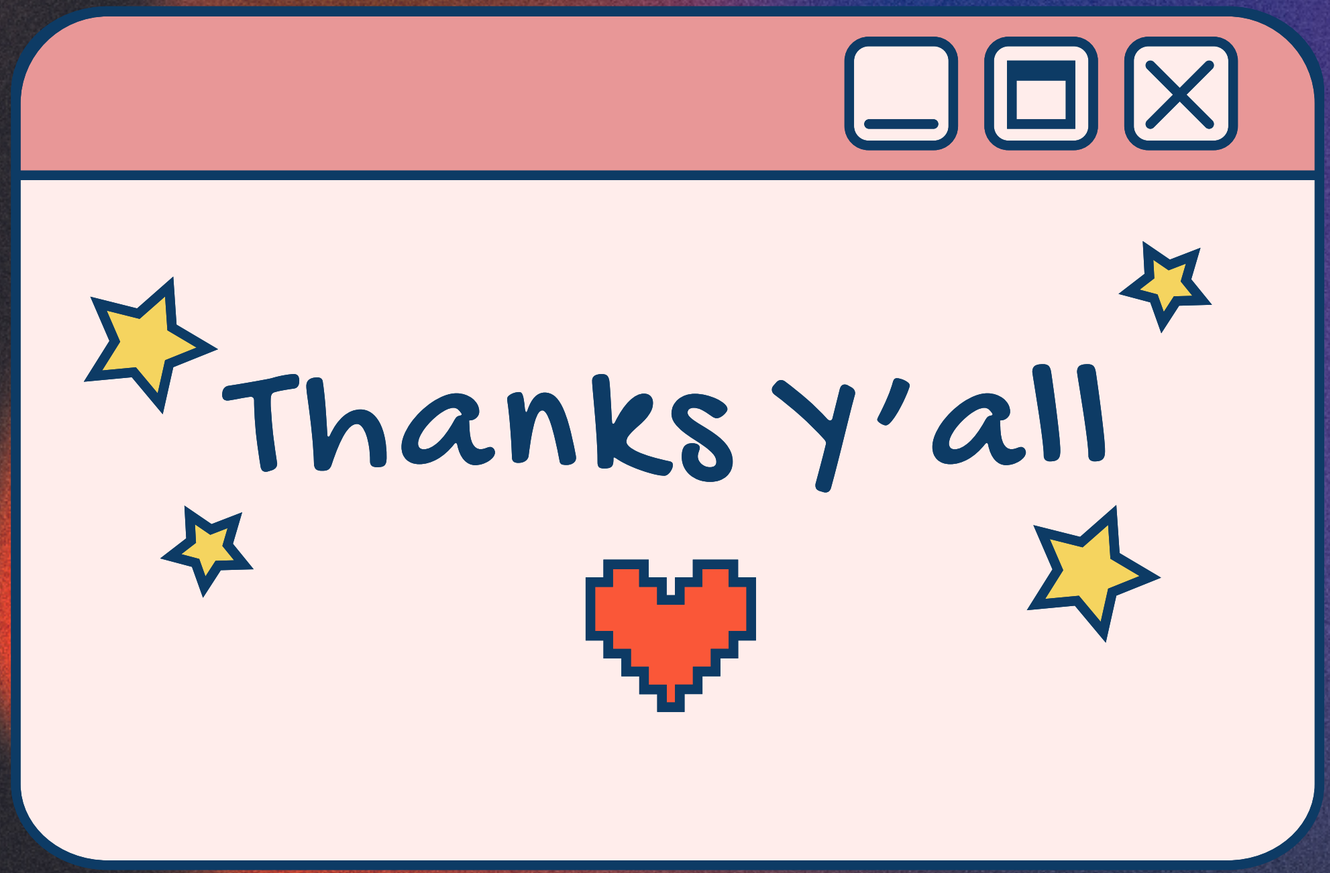
EVALUATION

EVALUATING THE MODEL

```
score = model.evaluate(X_train, Y_train, verbose=0)
print("Training Accuracy: ", score[1]*100)

score = model.evaluate(X_test, Y_test, verbose=0)
print("Testing Accuracy: ", score[1]*100)
```

We are getting training accuracy of 89%.
And test accuracy of 75%.


Thanks Y'all