# APPENDIX A

# PROJECT CODE

## A.1 Model training and testing

```
#!/usr/bin/env python
# coding: utf-8
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')


df = pd.read_csv('input/train_post_competition.csv')
df.head()

from IPython.display import Audio
file = '786ee883.wav'
path = 'input/audio_train/'
Audio(filename=path+file)

import wave

def get_length(file):
    audio = wave.open(path+file)
    return audio.getnframes() / audio.getframerate()
```

```
get_length(file)


from joblib import Parallel, delayed

with Parallel(n_jobs=10, prefer='threads', verbose=1) as ex:
    lengths = ex(delayed(get_length)(e) for e in df.fname)


df['length'] = lengths
df.head()


df = df.query('length␣<=␣6').reset_index(drop=True)
print(df.shape)
df.head()

import librosa

y, sr = librosa.load(path+file)
# y : audio data
# sr: sample rate

plt.plot(y)
plt.title(f'Sample␣rate␣=␣{sr}', size=18);


mfcc = librosa.feature.mfcc(y, sr, n_mfcc=40)
print(mfcc.shape)

plt.figure(figsize=(10,5))
plt.imshow(mfcc, cmap='hot');
```

```
def obtain_mfcc(file, features=40):

    y, sr = librosa.load(path+file, res_type='kaiser_fast')
    return librosa.feature.mfcc(y, sr, n_mfcc=features)


obtain_mfcc(file).shape

mfcc.shape



def get_mfcc(file, n_mfcc=40, padding=None):

    y, sr = librosa.load(path+file, res_type='kaiser_fast')
    mfcc = librosa.feature.mfcc(y, sr, n_mfcc=n_mfcc)
    if padding: mfcc = np.pad(mfcc, ((0, 0), (0, max(0,
    padding-mfcc.shape[1]))), 'constant')
    return mfcc.astype(np.float32)



mfcc = get_mfcc(file, padding=200)
print(mfcc.shape)
plt.figure(figsize=(12,5))
plt.imshow(mfcc, cmap='hot');

print(get_mfcc(df.sort_values('length').fname.iloc[-1]).shape)
```

```
from functools import partial

n_mfcc = 40
padding = 259
fun = partial(get_mfcc, n_mfcc=n_mfcc, padding=padding)

with Parallel(n_jobs=10, prefer='threads', verbose=1) as ex:
    mfcc_data = ex(delayed(partial(fun))(e) for e in df.fname)


mfcc_data = np.stack(mfcc_data)[..., None]
mfcc_data.shape




lbl2idx = {lbl:idx for idx,lbl in enumerate(df.label.unique())}
idx2lbl = {idx:lbl for lbl,idx in lbl2idx.items()}
n_categories = len(lbl2idx)



n_categories = len(lbl2idx)

df['y'] = df.label.map(lbl2idx)
df.head()



from sklearn.model_selection import train_test_split

x_train, x_val, y_train, y_val = train_test_split(mfcc_data, df.y,
test_size=0.2, random_state=42)
x_train.shape, x_val.shape
```

```
from keras.models import Model
from keras.layers import Dense, Conv2D,
BatchNormalization, Dropout, Input, GlobalAvgPool2D,
GlobalMaxPool2D, concatenate
from keras.optimizers import Adam, SGD
import keras.backend as K



bs = 128
lr = 0.003

m_in = Input([n_mfcc, padding, 1])
x = BatchNormalization()(m_in)

layers = [10, 20, 50, 100]
for i,l in enumerate(layers):
    strides = 1 if i == 0 else (2,2)
    x = Conv2D(l, 3, strides=strides, activation='relu',
    padding='same',
                    use_bias=False,
                    kernel_initializer='he_uniform')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.02)(x)

x_avg = GlobalAvgPool2D()(x)
x_max = GlobalMaxPool2D()(x)

x = concatenate([x_avg, x_max])
x = Dense(1000, activation='relu', use_bias=False,
kernel_initializer='he_uniform')(x)
```

```
x = Dropout(0.2)(x)
m_out = Dense(n_categories, activation='softmax')(x)

model = Model(m_in, m_out)
model.compile(Adam(lr),
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.summary()

log1 = model.fit(x_train, y_train, bs, 15,
validation_data=[x_val, y_val])



K.eval(model.optimizer.lr.assign(lr/10))
log2 = model.fit(x_train, y_train, bs, 10,
validation_data=[x_val, y_val])




def show_results(*logs):
    trn_loss, val_loss, trn_acc, val_acc = [], [], [], []

    for log in logs:
        trn_loss += log.history['loss']
        val_loss += log.history['val_loss']
        trn_acc += log.history['acc']
        val_acc += log.history['val_acc']

    fig, axes = plt.subplots(1, 2, figsize=(14,4))
    ax1, ax2 = axes
    ax1.plot(trn_loss, label='train')
    ax1.plot(val_loss, label='validation')
    ax1.set_xlabel('epoch'); ax1.set_ylabel('loss')
    ax2.plot(trn_acc, label='train')
    ax2.plot(val_acc, label='validation')
    ax2.set_xlabel('epoch'); ax2.set_ylabel('accuracy')
    for ax,title in zip(axes, ['Train', 'Accuracy']):
        ax.set_title(title, size=14)
        ax.legend()



show_results(log1, log2)



sample = df.sample()
sample_file = sample.fname.iloc[0]
sample_label = sample.label.iloc[0]

mfcc = get_mfcc(sample_file, n_mfcc, padding)[None, ..., None]
y_ = model.predict(mfcc)
pred = idx2lbl[np.argmax(y_)]



print(f'True␣␣␣␣␣␣␣=␣{sample_label}')
print(f'Prediction␣=␣{pred}')
Audio(path + sample_file)



def get_mfcc2(file, n_mfcc=40, padding=None):
    y, sr = librosa.load(file, res_type='kaiser_fast')
    mfcc = librosa.feature.mfcc(y, sr, n_mfcc=n_mfcc)
    if padding: mfcc = np.pad(mfcc, ((0, 0), (0, max(0,
    padding-mfcc.shape[1]))), 'constant')
    return mfcc.astype(np.float32)

mfcc = get_mfcc2("test_audio.wav", n_mfcc, padding)[None,
..., None]
y_ = model.predict(mfcc)
pred = idx2lbl[np.argmax(y_)]
print(pred)

model.save('best_model.h5')



from keras.models import load_model
model = load_model('best_model.h5')



import librosa
n_mfcc = 40
padding = 259
mfcc = get_mfcc("047b3d34.wav", n_mfcc, padding)[None,
..., None]
y_ = model.predict(mfcc)
pred = idx2lbl[np.argmax(y_)]
print(pred)
```

## A.2 Audio classification web classification

```
from flask import Flask,render_template,
flash, url_for, request,redirect,Blueprint
from flask import Response, make_response
import requests
import json
import sys, os
import math
import numpy as np
import pandas as pd
import wave
import librosa

from keras.models import Model
from keras.layers import Dense, Conv2D, BatchNormalization, Dropout, Inpu
from keras.optimizers import Adam, SGD
import keras.backend as K

from keras.models import load_model

import pymongo
from pymongo import MongoClient
mongoClient = MongoClient('localhost',27017)
db=mongoClient['coughTracker']
user_collection=db.users

bs = 128
lr = 0.003

df = pd.read_csv('input/
train_post_competition.csv')
```

```
def obtain_mfcc(file, features=40):
    y, sr = librosa.load(path+file, res_type='kaiser_fast')
    return librosa.feature.mfcc(y, sr, n_mfcc=features)

def get_mfcc(file, n_mfcc=40, padding=None):
    y, sr = librosa.load(file, res_type='kaiser_fast')
    mfcc = librosa.feature.mfcc(y, sr, n_mfcc=n_mfcc)
    if padding: mfcc = np.pad(mfcc, ((0, 0), (0, max(0, padding-mfcc.shap
    return mfcc.astype(np.float32)

app = Flask(__name__)
app.secret_key='asdasd^%$%^&asdjh%^$f^'
@app.route('/login')
def login():
    return render_template("login.html")

@app.route('/logout')
def logout():
    resp = make_response(redirect(url_for('login')))
    resp.set_cookie('clientId','', expires=0)
    return resp

@app.route('/loginVerify',methods=['GET', 'POST'])
def loginVerify():
    clientId = request.form['clientId']
    resp = make_response(redirect(url_for('index')))
    resp.set_cookie('clientId', clientId,max_age=60*60*12)
    return resp

@app.route('/index')
def index():
    if request.cookies.get('clientId') is not None:
        clientId = request.cookies.get('clientId')
        print("Inside␣index")
        print(clientId)
        if user_collection.find_one({"clientId":clientId}) is not None:
            coughCount = user_collection.find_one({"clientId":clientId})[
            print("inside␣if␣of␣index")
            print(coughCount)
            if coughCount > 3 :
                return render_template('index.html', clientId=clientId, c
            else:
                return render_template('index.html', clientId=clientId, c
        else:
            data = {"clientId":clientId,"coughCount":0}
            user_collection.insert_one(data)
            print("New␣client␣created")
            return render_template('index.html', clientId=clientId, cough
    else:
        return redirect(url_for('login'))

@app.route('/saveSound',methods=['GET', 'POST'])
def saveSound():
    data = request.data
    print("hello")
    #print(data)
    with open("test_audio.wav","wb") as fo:
        fo.write(data)
    print(request)
    return Response("{'a':'b'}", status=201, mimetype='application/json')

@app.route('/audioClassify',methods=['GET', 'POST'])
def audioClassify():
    model = load_model('best_model.h5')
    model._make_predict_function()
    n_mfcc = 40
    padding = 259
    mfcc = get_mfcc("test_audio.wav", n_mfcc, padding)[None, ..., None]
    y_ = model.predict(mfcc)
    pred = idx2lbl[np.argmax(y_)]
    print(pred)
    clientId = request.cookies.get('clientId')
    print("Inside␣audioClassify")
    print(clientId)
    if pred == "Cough":
        coughCount = user_collection.find_one({"clientId":clientId})["cou
        print("current␣coughCount")
        print(coughCount)
        print("coughCount+1")
        coughCount=coughCount+1
        print("New␣coughCount:")
        print(coughCount)
        user_collection.update_one({"clientId":clientId},{"$set": { "coug

    flash("Sound␣is␣:"+pred)
    K.clear_session()
    os.system("rm␣-rvf␣test_audio.wav")
    #return render_template('index.html')
    return redirect(url_for('index'))


if __name__ == '__main__':
    context = ('ssl.cert', 'ssl.key')
    app.run(host='0.0.0.0',port=8124,ssl_context=context)
```