HOW TO DEPLOY THE ML MODEL ?

FAQ

- **What is Cloud Functions ?**

  Cloud Functions is a managed service for serverless functions. The acronym describing such a service is FaaS (Function as a Service).

- **What's a managed service?**

  It's a service I don't have to manage. I just use it.

- **What's a serverless function?**

  Like a function in a program, a serverless (or cloud) function is an independent unit that can be naturally isolated in an app architecture.

- **How does it work?**

  The function is event driven: code will be called upon triggers such as an HTTP request, a file uploaded to a cloud storage, a message published...

- **What are the benefits of a serverless function?**

  The code itself can be directly deployed in an isolated cloud function.

  The function resources scale automatically, up but also down, as the traffic evolves (the infra handles this for me).

  The function is highly available (a benefit of using a managed service).

  I only pay for what I use: if the function is not used, the cost is zero.

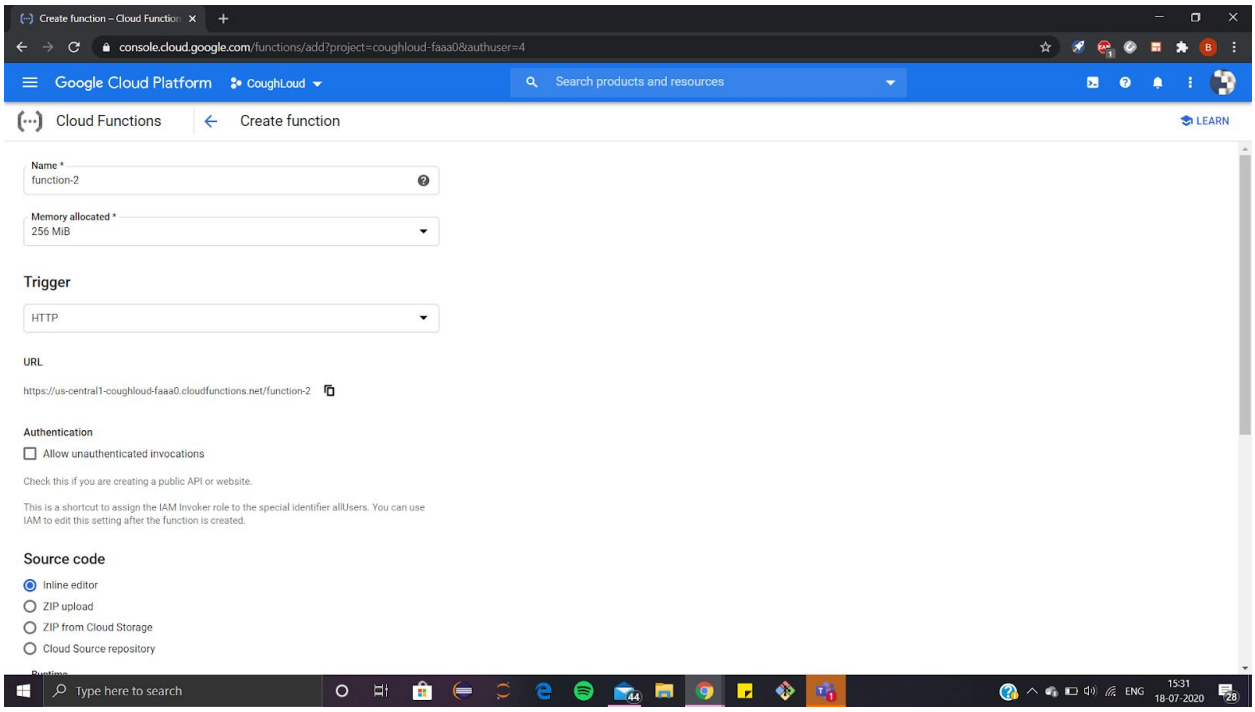
OVERVIEW : COUGH CLASSIFIER FOR COUGHLOUD

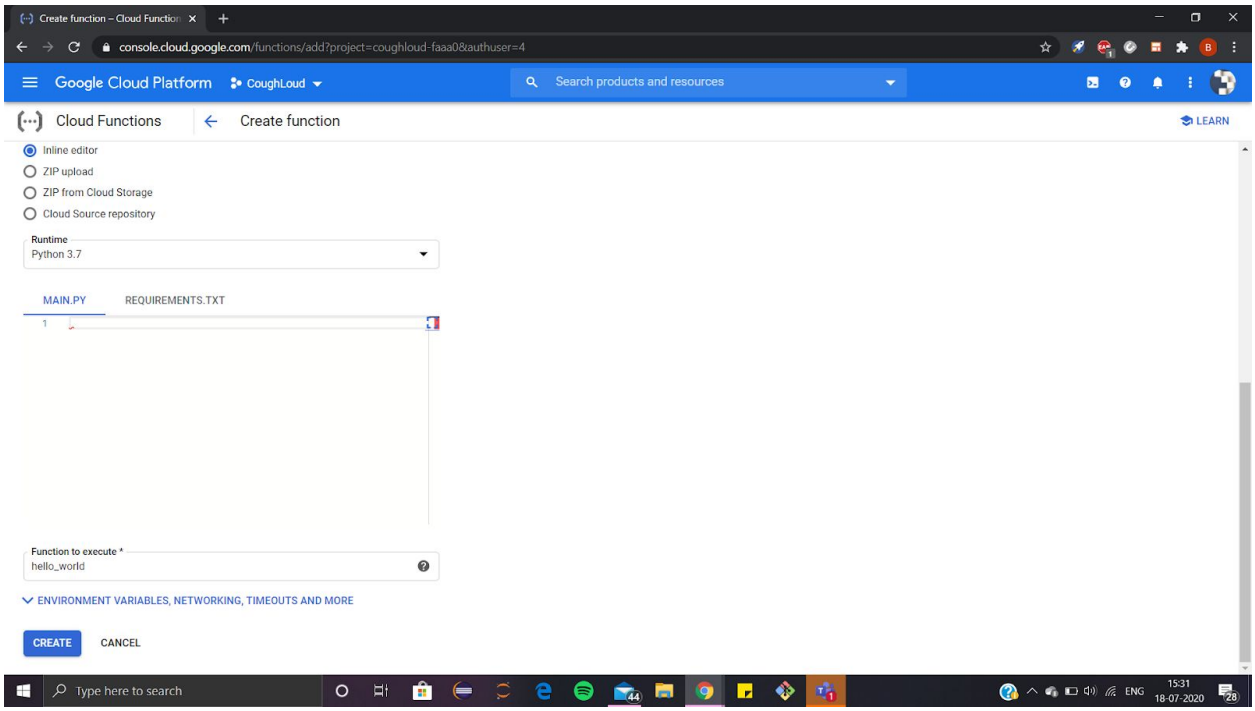
We Deployed a cloud function from the GUI to make our cough-classifier model come into live action. Cloud Functions are used because of the various benefits it provides(as mentioned above).

In this documentation we will talk about how we deployed our model and various technicalities involved in it, So let's get started.

CREATING A CLOUD FUNCTION

First we created a project name CoughLoud from the web console and then created a cloud function in the same project, Next step was to give some basic details as mentioned below





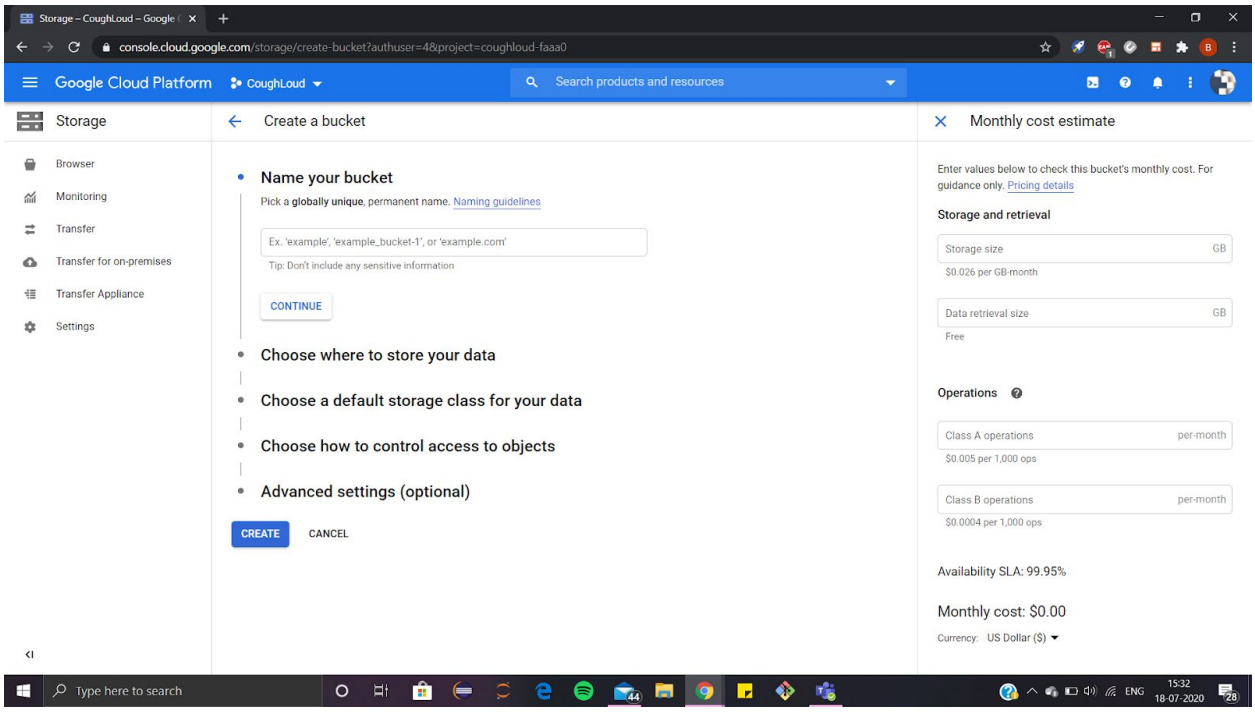
- Name - here we have to write our function name, example6 in our case
- Memory allocated- Cloud Functions provides access to a local disk mount point (/tmp) which is known as a "tmpfs" volume in which data written to the volume is stored in memory. We can allocate the memory varying from 128 Mib to 2Gib, We used 1Gib.
- Trigger - Triggers connect events to Cloud Functions. Events happen irrespective of triggers and the existence of functions. But it is the trigger that maps an event to a function, In our case we used HTTP trigger and after selecting HTTP trigger you will also get the URL for the same in the next line.
- Source code - We used inline editor as our Source code, (you can select according to you code)
- Runtime - Python 3.7, (you can select according to you code)
- Advance options - We used uscentral1 region and timeout of 540 seconds.
- Function to execute- Write the name of the function which you want to execute from your main.py

Two files namely main.py and requirements.txt are created after selecting the runtime Python 3.7, Where we have to write our code in main.py and the packages required for the code are mentioned in requirements.txt.

We will discuss about technicalities of main.py later, let's first discuss about creating a bucket

CREATING A BUCKET

We can create a bucket in storage section of web-console.



All default options was selected.

The recorded coughs are saved in our bucket from where we access the audio file and predict that it is a dry or wet cough using our cloud function.

Now we are all set to discuss main.py.

MAIN.PY

We uploaded the required file (model and dataset file) in the bucket where coughs are stored. So that after importing google cloud storage and using blob we can access that files to load our model and for the preprocessing steps.

We first linked the required bucket using

```
bucket = storage_client.get_bucket('[bucket name]')
```

Then by using

```
blob = bucket.blob([file location])
```

we stored the required file in blob and later downloaded it to local disk(/tmp/) using

```
blob.download_to_filename(f'/tmp/{filename}')
```

from where we can access it whenever required like for example for loading the model we use

```
new_model = tf.keras.models.load_model('/tmp/ANN_model')
```

Now in the function which is to be executed we first access the file using the blob concept mentioned above and the required preprocessing was done to get an array which is used by our model to predict the result.

A class object was created with variable ‘cough’ and ‘per’ denoting type of the cough and the accuracy with which our model predicts that cough respectively.

By our model result these values are passed to the class object, which later is used to create python dictionary as follow

dict = {

"type" : p.cough,

"percentage" : p.per.__str__() } , where p is our class object.

If per < 80% we set the cough = 'Normal Cough' , to be more precise about result.

In the last step this dict is converted to json format and returned.

Requirements.txt

Following packages were used for the code in main.py

- numpy==1.18.5
- scikit-learn==0.23.1
- pandas==1.0.5
- librosa==0.7.2
- numba==0.48**
- tensorflow==2.1.0
- keras==2.3.1
- google-cloud-storage==1.17.0

**seems like numba removed the decorators module with version 0.50. To use numba decorators version 0.48 was installed

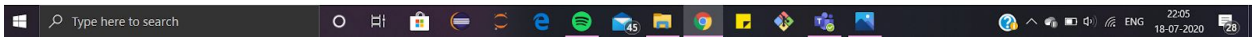
***The code is available at - https://github.com/avakar-healthwin/CoughClassifier.git*

DEPLOYMENT AND TESTING

After creating CLoud Function, we have to write a function to be executed and click on the deploy button given below.

The function deployed required the file name which is given with URL as Url + ?name=filename ,example - *https://us-central1-coughloud-faaa0.cloudfunctions.net/example5?name=yo.wav*

When you click on this Url will see the following result



For testing this function various filename was tried by different users which was present in the bucket storage.

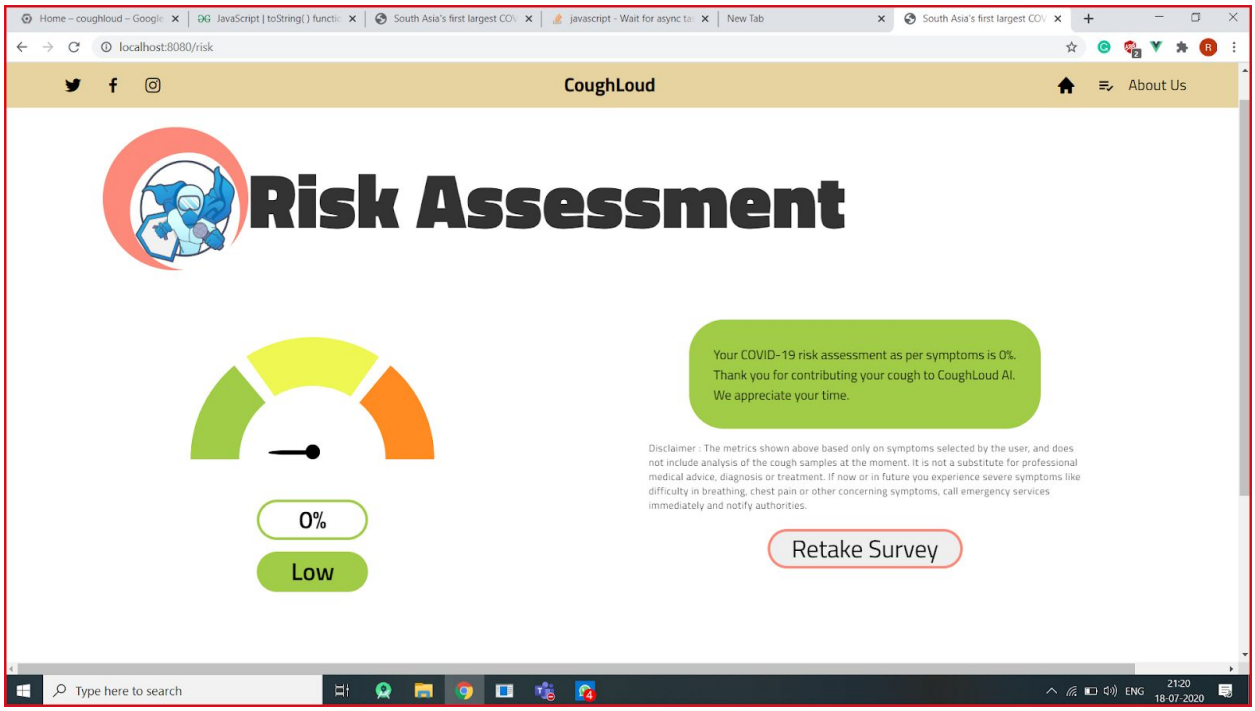
Time taken by function to execute when you check the result for a file through URL was also looked in Logs Viewer, Some results are shown below -

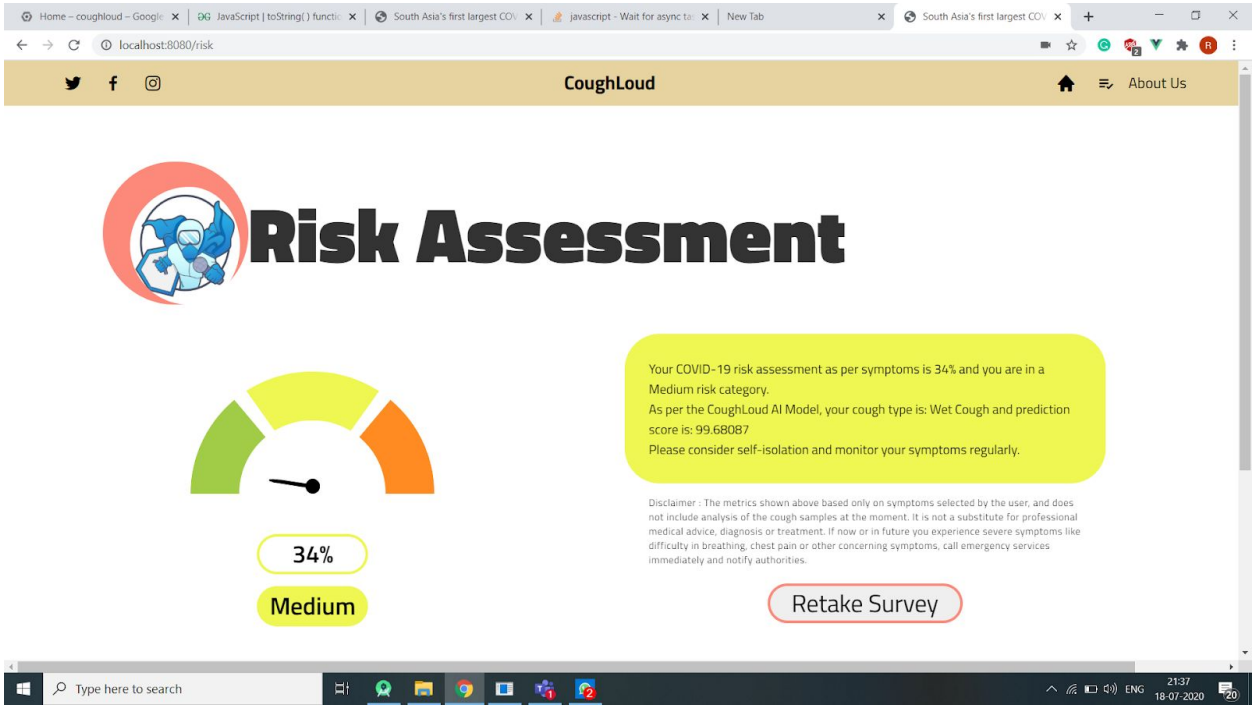
INTEGRATION TO COUGHLOUD PROJECT

Coughloud deep learning model is hosted on google cloud functions as http trigger and Coughloud project is made in vuejs framework. In vuejs, there is an 'axios' library for calling https functions. Axios allow us to call the google cloud http function. Keep in mind to call the axios when there is state_changed (trigger in cloud storage) in the function. For calling the http function, pass the file name (e.g. '14594358af.wav') as url arguments. Since it is an asynchronous task, it returns a promise. That promise returns a response, and then saving that response data into an info object type variable. And at last, the variable is displayed on webpage.

```
}, function() {
    uploadTask.snapshot.ref.getDownloadURL().then(function(downloadURL) {
        vue.dUrl = downloadURL;
    //-----------google cloud function for ml model---------------------------------
        axios
        .get('https://us-central1-coughloud-faaa0.cloudfunctions.net/example5?name='+u+'.wav')
        .then((response) => {
            vue.info = response.data;
```

RESULT PAGE





***For 0% risk, we didn't show the prediction of our ML model.*

TROUBLESHOOTING

CORS ERROR - A request for a resource (like an image or a font) outside of the origin is known as a cross-origin request. CORS (cross-origin resource sharing) manages cross-origin requests. The CORS standard is needed because it allows servers to specify not just who can access its assets, but also how the assets can be accessed. Therefore to allow '*Access-Control-Allow-Origin*' the following code was written -

```
if request.method == 'OPTIONS' :

    headers = {

        'Access-Control-Allow-Origin' : 'http://coughloud.healthwin.in',

        'Access-Control-Allow-Methods' : 'POST',

        'Access-Control-Allow-Headers' : 'Content-Type',
```

```
            'Access-Control-Max-Age' : '3600'
        }
        return '',204,headers
```

.....

```
return y,200,headers
```

If function executed successfully it will finished with status code : 200

TRIGGER ISSUES - Several permissions were denied due to which error 403 was shown, by allowing unauthenticated in Authentication, the error was solved.

STORAGE ACCESS - When we directly tried to access bucket storage using *'gs://bucket-name/'* we got an error, Therefore bolb concept was used to troubleshoot this problem.

CREATE AN EMPTY BLOB - A file was required to upload in bucket storage but for that an empty blob was required, we tried several approaches from stackoverflow but they was unsuccessfull, So an alternate way of directly using the bucket in which file are stored was implemented.

EXECUTION TIME OF CLOUD FUNCTIONS - Due to short timeout session execution function can be crashed if it takes more time. Therefore to avoid this timeout limit was set to 540s for intial purposes which will later be set to particular threshold after doing some research on how much average time our function takes to execute.

PACKAGE ISSUES (Requirements) - The version of the package used should be noted properly as numba version 0.50 seems to have removed numba decorators, but we required it in our code, So version 0.48 was installed. Similarly exact version of packages was mentioned to avoid any incompatibility due to different versions.

NEXT STEPS

In future when the model improves (by training on more data and better feature extraction etc) we can replace our current model with the new version of the model by simply

replacing the current *ANN_model* file with the new one. This will work in case of if the new version is also ANN type.

If the new model is CNN or RNN type then also their replacing model is quite easy as you just have to change the preprocessing steps and replace the current model file with a newer one.

Modularising - We can divide our main.py into several cloud functions like one for preprocessing,one for loading the model and last one for predicting the result, doing this will reduce the the time taken by function to execute as we don't have to do whole thing in one function again and again.

Run times limit settings - We can set the timeout variable, If function takes more time to execute than the timeout, the session will crash.

Monitoring the function - In the general section of our cloud function there is a graph showing how many times our function *crashed* and performed *ok* , Also you can view the *Logs Viewer* for more details. So monitoring of function is quite easy as google cloud function provides very useful tools to monitor our deployed function .Also If any update comes to your model or you get better feature extraction steps etc then you can make changes to your function very easily.

REFERENCES

https://www.codecademy.com/articles/what-is-cors

https://mp4-a.udemycdn.com/2020-02-12_03-47-40-09268e3b675ecf213c89dc11a1fffd6c/WebHD_720p.mp4?rHVBm0bXIxXlPRzU_J9FTtpHIY068aXTua8ux1r_adQPUFPJzXwm_TpnJVhENcMmgOr1TLj3SRb2zxFzXp1k3yD5U0WL8Xfi8A0tNQLiBitK6yE2pwA8U8YtA2CTeouRjMdOcCjNe3ziF2O9i4R8JqfSXrtbSBx2wyq20qILxADYSA

https://googleapis.dev/python/storage/latest/blobs.html

https://medium.com/google-cloud/deploying-a-python-serverless-function-in-minutes-with-gcp-19dd07e19824