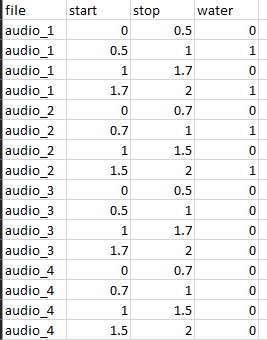
# Challenge

* Determine the time stamp of “tap water” sound in the given audio file
* Context:
  + This is internal company hackathon. Will be started in a few days when I wrote this but I started exploring earlier :D (My motivation: learning and networking to switch role to “data” job )
  + To determine how much water is consumed during the use of our product. Related to sustainability
* Sample dataset:
  + Audio files (various format, bitrate, #channel)
  + CSV file detailing the timestamp of “tap water” sound in each audio files. Example:
    - 
* Test dataset:
  + Audio files (various format, bitrate, #channel)
  + Expected output:
    - CSV file detailing the timestamp of “tap water” sound in each audio files

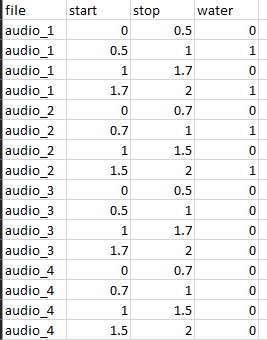
# Solution Overview

NOTE: Below solution has a lot of improvement points. Important thing for me is the concept is proven works.

Idea is as [this](https://www.tensorflow.org/tutorials/audio/simple_audio) Tensorflow tutorial (google keyword: “audio classification tensorflow”). We would tailor it to match our challenge:

* Sample dataset and label should be like below:
  + ‘water’:
    - sample of water sound.
    - Each file is X seconds in length. (I am thinking 0.5 second)
    - Must be wav file
    - Must be in 16000 sampling rates
    - Must be in 1 channel
  + ‘non\_water’ : Same as above, however this is for non-water sound sample.
* Real/Test dataset:
  + Same as above.
* This becomes a binary classification model. The model should determine whether the test dataset is water or non\_water.
* The model would be saved, so it can run any other test dataset that will be provided in a later time

However, the dataset provided in our challenge is different. So, a preprocessing and postprocessing is needed.

* Dataset provided:
  + Lengthy audio file, combination of water and non\_water sound in each file. Each file comes with a csv file explaining the detail timeline.
    - 
  + The audio format varies (most likely not only wav, but also mp3 or other)
  + Different sampling rate (16k, 44.1k, 48k)
  + Different # of channel (1 or 2)
* Preprocessing:
  + For the sample dataset:
    - Split the audio file to water and non\_water to be the dataset form that we want.
    - For example, audio\_1 would become as below (assuming we chose split length X = 0.5):
      * filename – row in CSV file – split number – water or non water
      * audio\_1\_1\_1\_non\_water 🡨 0.0 – 0.5 second
      * audio\_1\_1\_2\_non\_water 🡨 0.5 – 1.0 second
      * audio\_1\_1\_3\_non\_water 🡨 1.0 – 1.5 second
      * audio\_1\_1\_4\_non\_water 🡨 1.5 – 2.0 second
      * audio\_1\_1\_5\_non\_water 🡨 1.5 – 2.431 second
      * audio\_1\_2\_1\_water 🡨 2.431 – (2.431+0.5) second
      * audio\_1\_2\_2\_water 🡨 (2.431+0.5) – (2.431+0.5\*2) second
      * …
    - The name does not really matter. The split files should be organized in the right folder accordingly (folder name: water and non\_water)
  + For the real/test dataset:
    - Create a table to document this and save in a csv file for later reference
    - Split the audio file to X second
    - For example, test\_1 (length 1.71 seconds) would become as below (assuming we chose split length X = 0.5):
      * test\_1\_1 🡨 0.0 – 0.5 second
      * test\_1\_2 🡨 0.5 – 1.0 second
      * test\_1\_3 🡨 1.0 – 1.5 second
      * test\_1\_4 🡨 1.5 – 1.71 second
    - Table would look like:

|  |  |  |  |
| --- | --- | --- | --- |
| file  (original file name) | split  (split file name) | start  (start time) | stop  (end time) |
| test\_1 | test\_1\_1 | 0.0 | 0.5 |
| test\_1 | test\_1\_2 | 0.5 | 1.0 |
| test\_1 | test\_1\_3 | 0.1 | 1.5 |
| test\_1 | test\_1\_4 | 1.5 | 1.71 |

* + - * This is to be used in later reference
* Postprocessing
  + The output from model would be something like this:

|  |  |
| --- | --- |
| Split | Water  (1 : water; 0 : non\_water) |
| test\_1\_1 | 0 |
| test\_1\_2 | 1 |
| test\_1\_3 | 1 |
| test\_1\_4 | 0 |

* + It would be merged with original table with “split” as a key to create table as below

|  |  |  |  |
| --- | --- | --- | --- |
| file | start | stop | Water |
| test\_1 | 0.0 | 0.5 | 0 |
| test\_1 | 0.5 | 1.0 | 1 |
| test\_1 | 0.1 | 1.5 | 1 |
| test\_1 | 1.5 | 1.71 | 0 |

* + And then further process the table to group the row with consecutive ‘0’ or ‘1’ in column water

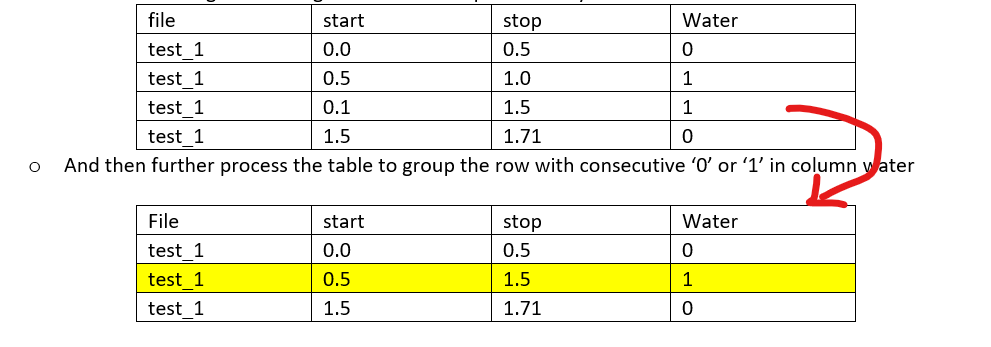
|  |  |  |  |
| --- | --- | --- | --- |
| File | start | stop | Water |
| test\_1 | 0.0 | 0.5 | 0 |
| test\_1 | 0.5 | 1.5 | 1 |
| test\_1 | 1.5 | 1.71 | 0 |

* + - See second row is combined with third row

# Key Learning

* The use of Pydub to work with audio. It’s powerful. It took me some time to realize that ffmpeg and other should be put in the same folder with the script
* I learned a lot in creating deep learning model with audio. The external courses that I took covered only the picture.

# Improvement point

* Rather than splitting the test dataset as preprocessing, probably better to just stride this directly in the audio file during processing
* Currently the window chosen is 0.5 second. Probably 0.2 or 0.3 could work too for better precision. I am reluctant to use a smaller window, I guess it has overfit risk.
* This may be small things, however I haven’t found a way to do below in a more Pythonic way. My approach is a bit crude
  + 
* There may be a better deep learning model. We can still play around with the node, layer, CNN, etc to find a better model.