PREDICTING AN EVENT FROM AUDIO DATA:

For my project I'd like to develop an algorithm to predict an event from audio data.

Specifically: within some margin of error, I'd like to be able to predict if a door has closed based on the sound a door makes when it closes.

PROJECT STEPS:

1) Locate data set
2) Read the data into Python
3) ‘Clean’ the data with FFT and/or other audio analysis tools
4) Divide the data set up into training and test
5) Perform unsupervised analysis to try to define what a cluster of similarly “shaped” data looks like with respect to frequency, time, and potentially amplitude or other variables
6) Perform supervised, categorical analysis to divide the data into a series of groups based on where in the data set the previously defined ‘shape’ fits, and what the margin of error for the fit is.
7) See how well the events actually predict the door closing

STEP 1) LOCATE THE DATA

I've found a 3-hour youtube video of somebody recording themselves repeatedly opening and closing a door:

https://www.youtube.com/watch?v=tULM0KBx7i8

(same guy that did the video of that waffle falling)

STEP 2) READ THE DATA INTO PYTHON

I've decided to use a python based command-line program called youtube-dl to download the audio from the youtube video.
(The github of youttube-dl can be found here: https://github.com/rg3/youtube-dl)

I've decided to use this program for several reasons:

1) File size: the previous strategy I had involved a website that can download a youtube video, however due to the length of this video predicted file size (6 GB) is too much for my computer

2) Efficiency: the terminal will be much more efficient than using a browser to download the data

3) More Control: I have more direct control over the quality of the file that is downloaded

Ideally I need this audio data as a .wav file or an .mp3 file to use the audio analysis tool I've found. However:

- The audio from the youtube video is only available for download as .webm or .m4a

- youtube-dl has a post-processing algorithm that can save the file as just about any file type I need, however I cannot get this to work yet as it requires the instillation of one of two additional programs to function (ffmpeg or avconv)

STEP 3) 'CLEAN' THE DATA

I've decided located a comprehensive open-source C++ audio analysis library called Essentia:

(Essentia can be found here: http://essentia.upf.edu/)

Essentia includes a collection of algorithms designed around extracting all sorts of features from audio files. The library also includes Python bindings, comprehensive documentation, and even a Python tutorial.

However:

- In order to use this program I need to get it to work (I haven't been able to get it to work yet)
- I don't think it can read a .webm file (possibly it can read an .m4a file)

Using Essentia I can easily perform many advanced audio analysis techniques including FFT (Fast Fourier Transform,) MFC (mel-frequency cepstrum,) and MFCC (mel-frequency cepstral coefficients) as well as pretty much anything else I can think of.

STEP 4) DIVIDE DATA INTO ‘TRAINING’ AND ‘TEST’

Fairly self-explanatory. I have 3 hours of data so I have plenty to work with. I’ll just use different time selections.

STEP 5) UNSUPERVISED ANALYSIS

Once I have the data in a more workable form I can try to do an analysis looking for patterns in the frequency over some time neighborhood.

Ideally I’d like to look for more general patterns and let the ‘shapes’ define themselves.

Variables I’d like to consider include frequency, time, amplitude, etc..

STEP 6) SUPERVISED ANALYSIS

Once I have shape patterns defined, I try to categorize the sound data into sections where the data fits them, with a corresponding margin of error.

STEP 7) HOW WELL THIS PREDICTS THE EVENT

Now that I have divided the sound data into a series of events, I can go back and look at how well the supervised analysis was able to categorize the ‘door closing’ event. (Probably I’ll have a ‘door not closing’ event as well)

Additionally I can try to define the margin of error at which I can accurately predict the event of the door closing.

SUPER REACH STEPS:

A. Write a program to predict the likeliness of a door closing based on a live audio input.