Table of Contents

[SCOPE 2](#_Toc514076262)

[BACKGROUND 2](#_Toc514076263)

[TensorFlow 2](#_Toc514076264)

[Anaconda Navigator 2](#_Toc514076265)

[ENVIRONMENT SETUP 2](#_Toc514076266)

[1. Anaconda Installation 2](#_Toc514076267)

[2. Tensorflow Installation 3](#_Toc514076268)

[3. Creating Tensorflow Environment in Anaconda 3](#_Toc514076269)

[4. Validation of Tensorflow Installation 6](#_Toc514076270)

[DATA EXPLORATION 8](#_Toc514076271)

[1. Data Examination 8](#_Toc514076272)

[Number of records 8](#_Toc514076273)

[2. Data Visualization 10](#_Toc514076274)

[Spectorgram 10](#_Toc514076275)

[MFCC 11](#_Toc514076276)

# SCOPE

The scope of the project to learn and understand TensorFlow and to use this open source software library for simple audio recognition.

# BACKGROUND

Audio recognition is used in a lot of applications such as Alexa, Ok Google, Siri and so on. TensorFlow an open source voice interface tool as released the Speech Commands Datasets. It includes 65,000 one-second long utterances of 30 short words, by thousands of different people. The challenge is to use TensorFlow and take advantage of TensorFlow’s dataset to implement an algorithm for speech recognition.

## TensorFlow

TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

## Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface included in Anaconda that allows you to launch applications and easily manage conda packages, environments and channels without the need to use command line commands.

# ENVIRONMENT SETUP

## Anaconda Installation

To install Tensorflow with Anaconda, first install Anaconda on your system. Install python 3.6 version.

**Link:**

<https://www.anaconda.com/download/>

A screenshot of a cell phone

Description generated with very high confidence

## Tensorflow Installation

Once Anaconda is installed successfully, follow to below steps to install tensorflow

1. Create a conda environment named tensorflow by invoking the following command:

C:> conda create -n tensorflow pip python=3.5

1. Activate the conda environment by issuing the following command:

C:> activate tensorflow

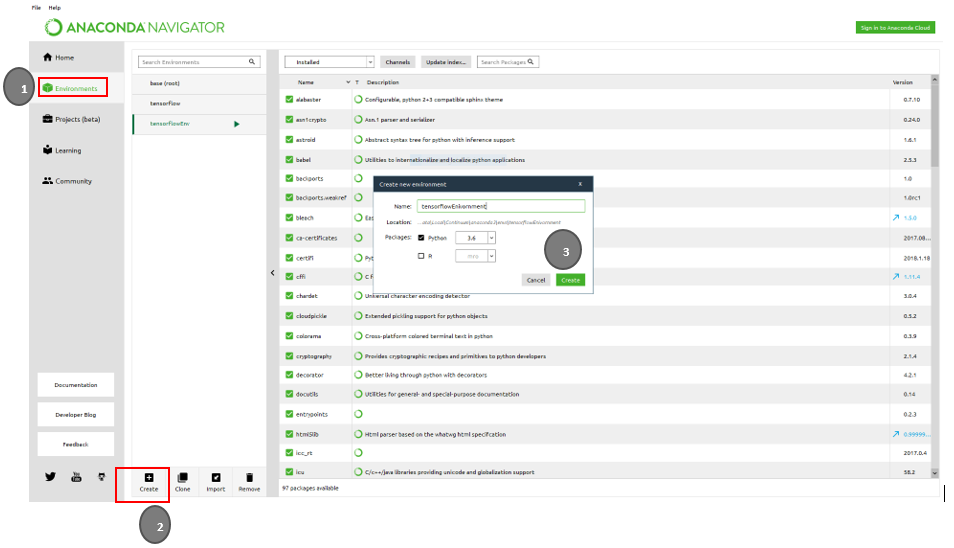
(tensorflow)C:> # Your prompt should change

1. Issue the appropriate command to install TensorFlow inside your conda environment. To install the CPU-only version of TensorFlow, enter the following command:

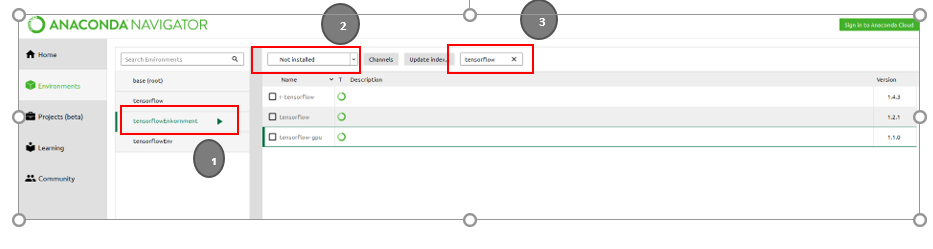
(tensorflow)C:> pip install --ignore-installed --upgrade tensorflow

## Creating Tensorflow Environment in Anaconda

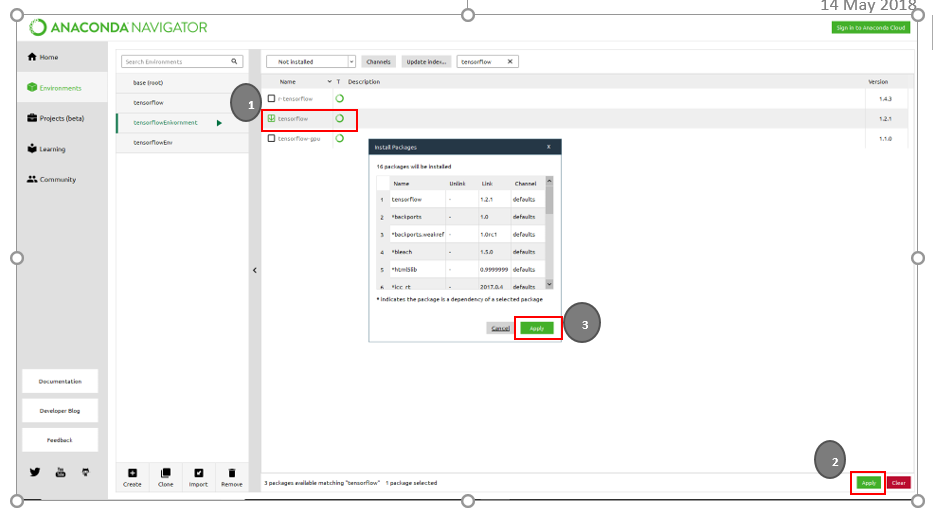
**Step 1:** open Anaconda navigator and click on environments, in environments click on create. This will open a dialogue box. Here name your environment and click on create.



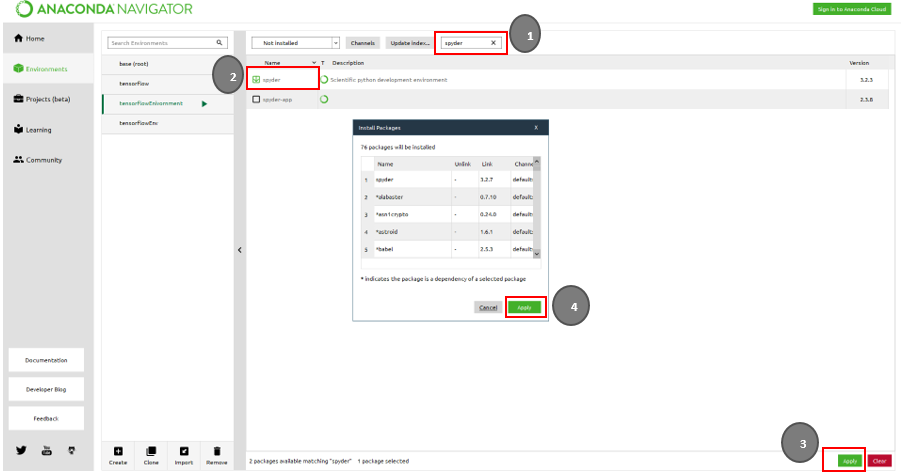
**Step 2:** Once the environment is created, click on your environment. you will see a list on installed packages. Now in the dropdown menu click on “not installed’ and in the search box type “tensorfow”. You will now see the three packages for tensorflow as shown in the screenshot.



**Step 3:** select tensorflow and click on apply. You will see a dialogue box with packages again click on apply, this will install tensorflow.

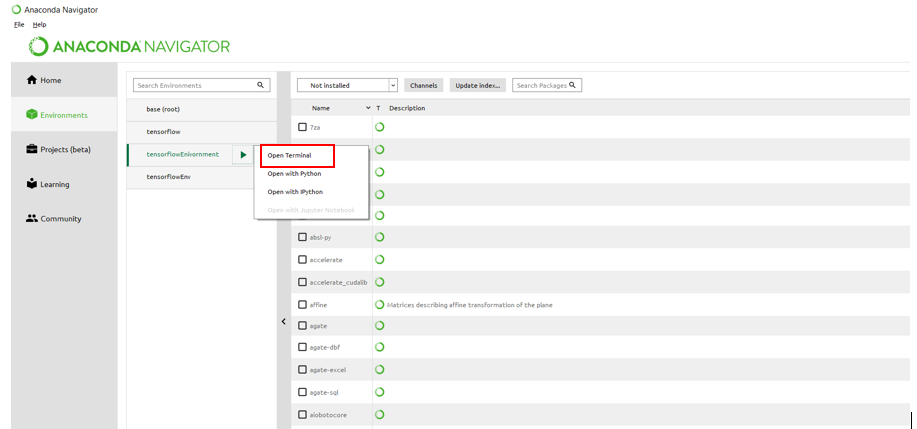


**Step 4:** In the search box type “spyder”, this will show two packages. Select spyder and click on apply. This will open another dialogue with the packages to be installed, click on apply again to install sypder.



## Validation of Tensorflow Installation

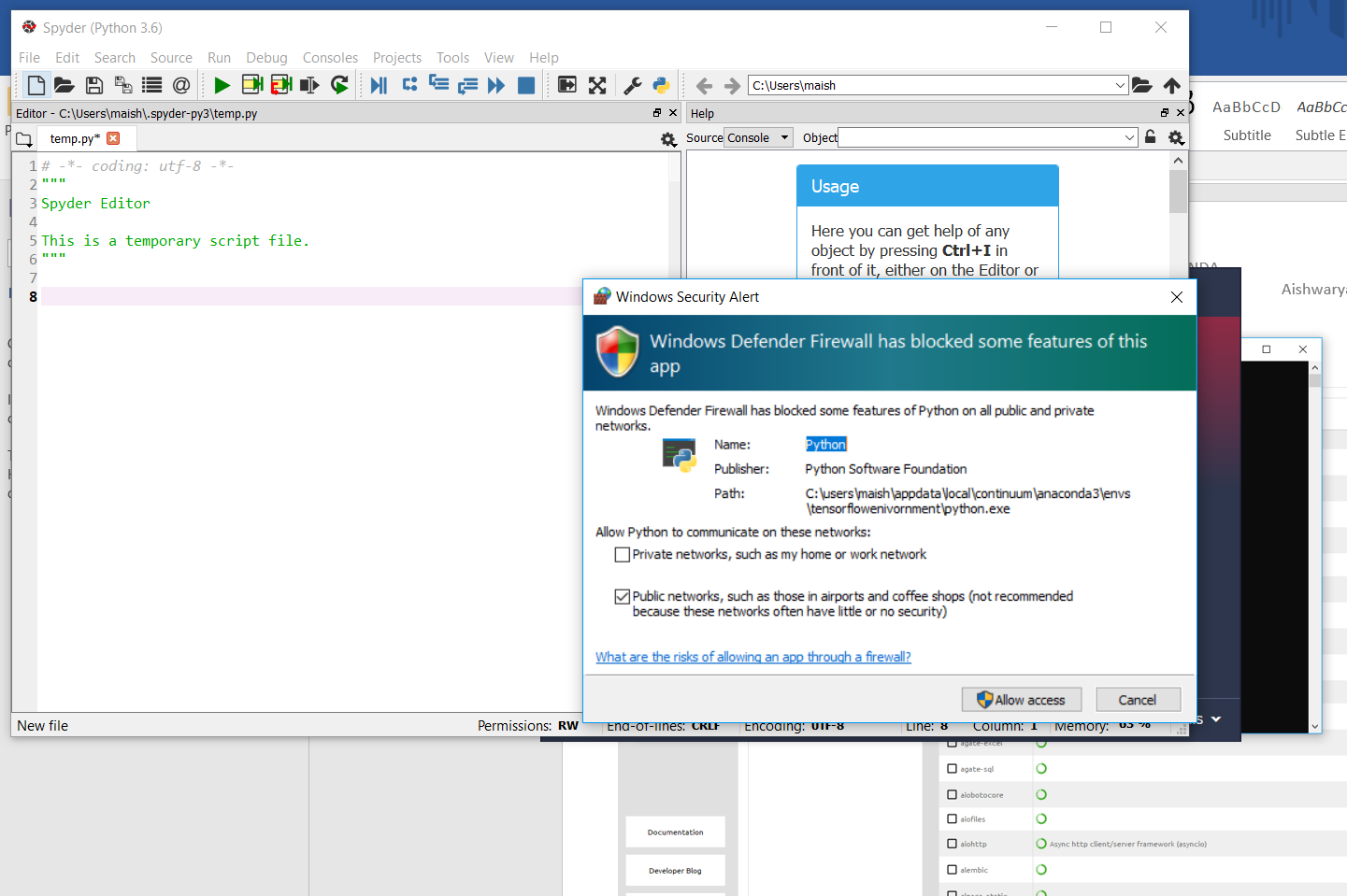
**Step 1:** right click on “tensorflowEnvironment” and select open terminal



**Step 2:** once the terminal is open, type “spyder” and press enter.



**Step 3:** once you press enter. Spyder window opens and you will get a security alert pop up. Click on “allow access”.



**Step 4:** To validate tensorflow installation**,** type the following code

**Code:**

>>> import tensorflow as tf  
>>> hello = tf.constant('Hello, TensorFlow!')  
>>> sess = tf.Session()  
>>> print(sess.run(hello))

If tensorflow is successfully installed, running the code should give you the following output

**Output:**

**'Hello, TensorFlow!'**



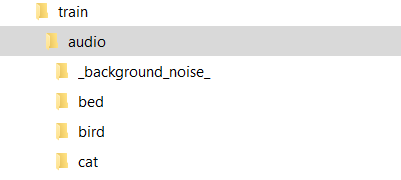
# DATA EXPLORATION

The following python packages were used for data exploration

* Matplotlib (A plotting library for the Python, its numerical mathematics extension NumPy)
* Librosa (A python package for music and audio analysis)
* Plotly (Python graphing library makes interactive, publication-quality graphs)

## Data Examination

The Folder structure is as shown in the screen shot below



Each of the folders under audio contains short utterances of the word in .wav files.

A simple investigation of the data shows that there is a total of 31 labels such ‘yes, no, up, down, left, right, stop ‘ and so on in the dataset.

**Code:**

dirs = [i for i in os.listdir(train\_audio\_path) if isdir(join(train\_audio\_path, i))]

dirs.sort()

print('Number of labels: ' + str(len(dirs)))

**Output:**

Number of labels: 31

Next, I wanted to check how many recordings are there under each label. For this I used the below code

**Code:**

number\_of\_recordings = []

for direct in dirs:

waves = [f for f in os.listdir(join(train\_audio\_path, direct)) if f.endswith('.wav')]

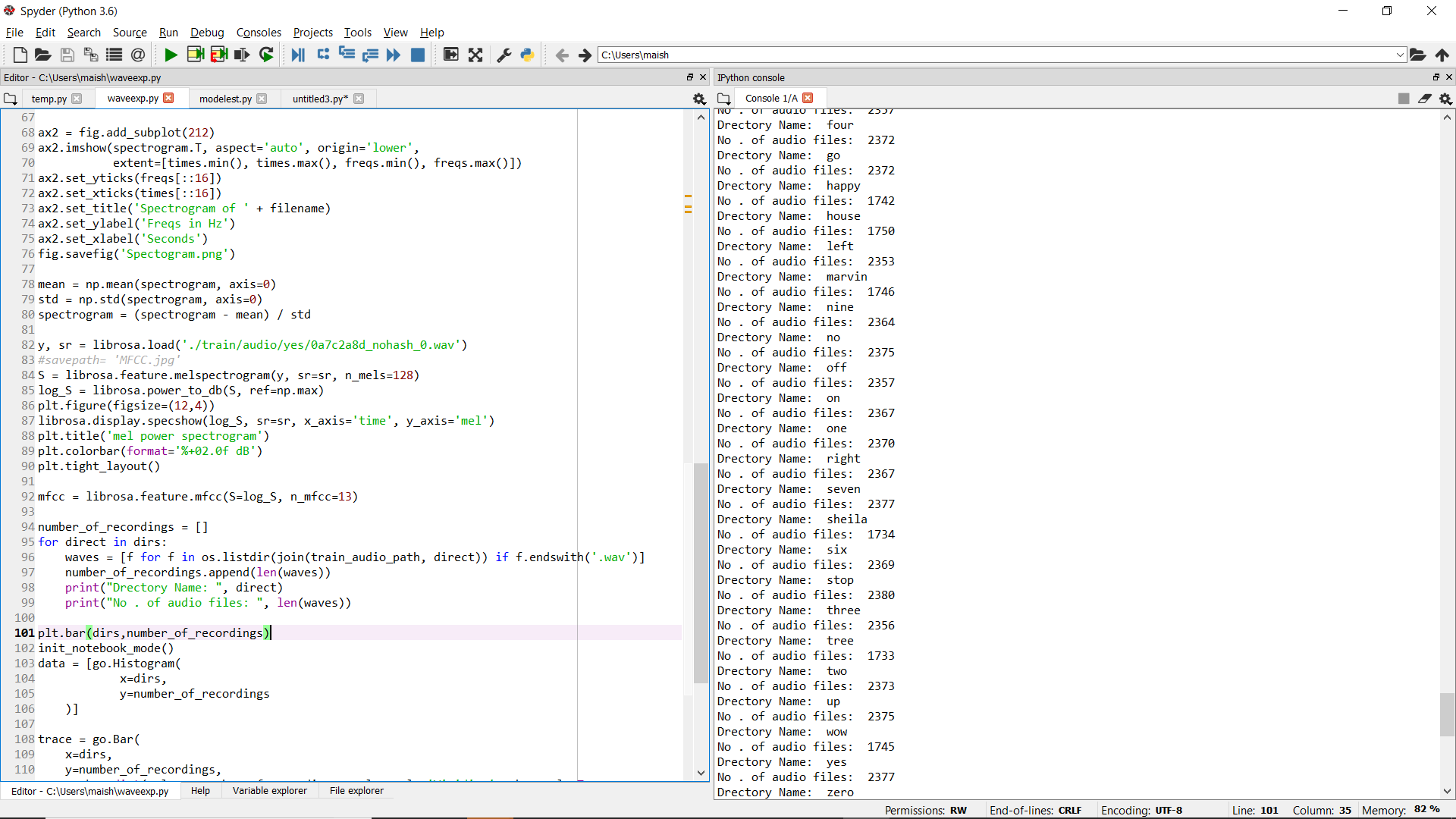
number\_of\_recordings.append(len(waves))

print("Drectory Name: ", direct)

print("No . of audio files: ", len(waves))

**Output:**

The output is displayed as shown in the screenshot below. The output shows the directory name and the number of recording in each of the directory.



To get a better visualization, I used matplotlib and plotly to graph the number of recordings in each label.

**Code:**

plt.bar(dirs,number\_of\_recordings)

init\_notebook\_mode()

data = [go.Histogram(

x=dirs,

y=number\_of\_recordings

)]

trace = go.Bar(

x=dirs,

y=number\_of\_recordings,

marker=dict(color = number\_of\_recordings, colorscale='Viridius', showscale=True

),

)

layout = go.Layout(

title='Number of recordings in given label',

xaxis = dict(title='Words'),

yaxis = dict(title='Number of recordings')

)

plot(go.Figure(data=[trace], layout=layout))

**Output:**

This generated the below plotly bar plot online.

A screenshot of a cell phone

Description generated with high confidence

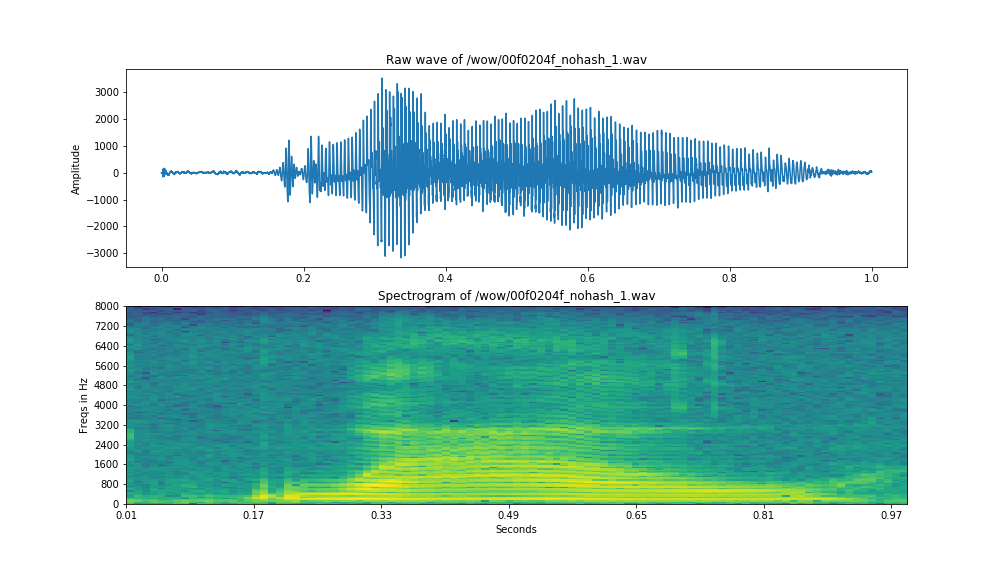
From the above chart, we can see that the recordings are evenly distributed, except for background noise.

## Data Visualization

Audio Files are visualized as spectrograms or MFCC - Mel-Frequency Cepstral Coefficients. Below we see the spectrogram and MFCC for “WOW”. Using Librosa package, we can visualize the MFCC and spectrogram of .wav file for audio WOW.

### Spectorgram

A spectrogram is a visual representation of the spectrum of frequencies of sound or other signal as they vary with time



### MFCC

MFCC are more sophisticated representation of audio and it closely replicated how the audio is heard by human ear. By converting the audio file to MFCC, the algorithm can better read the audio file.

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum.

The screen shot below shows the Mel Spectogram generated using librosa for the utterance WOW. The librosa code snippet used for generating Mel Spectogram spectrogram is as follows:

**Code:**

y, sr = librosa.load('./train/audio/wow/00f0204f\_nohash\_1.wav')

S = librosa.feature.melspectrogram(y, sr=sr, n\_mels=128)

log\_S = librosa.power\_to\_db(S, ref=np.max)

plt.figure(figsize=(12,4))

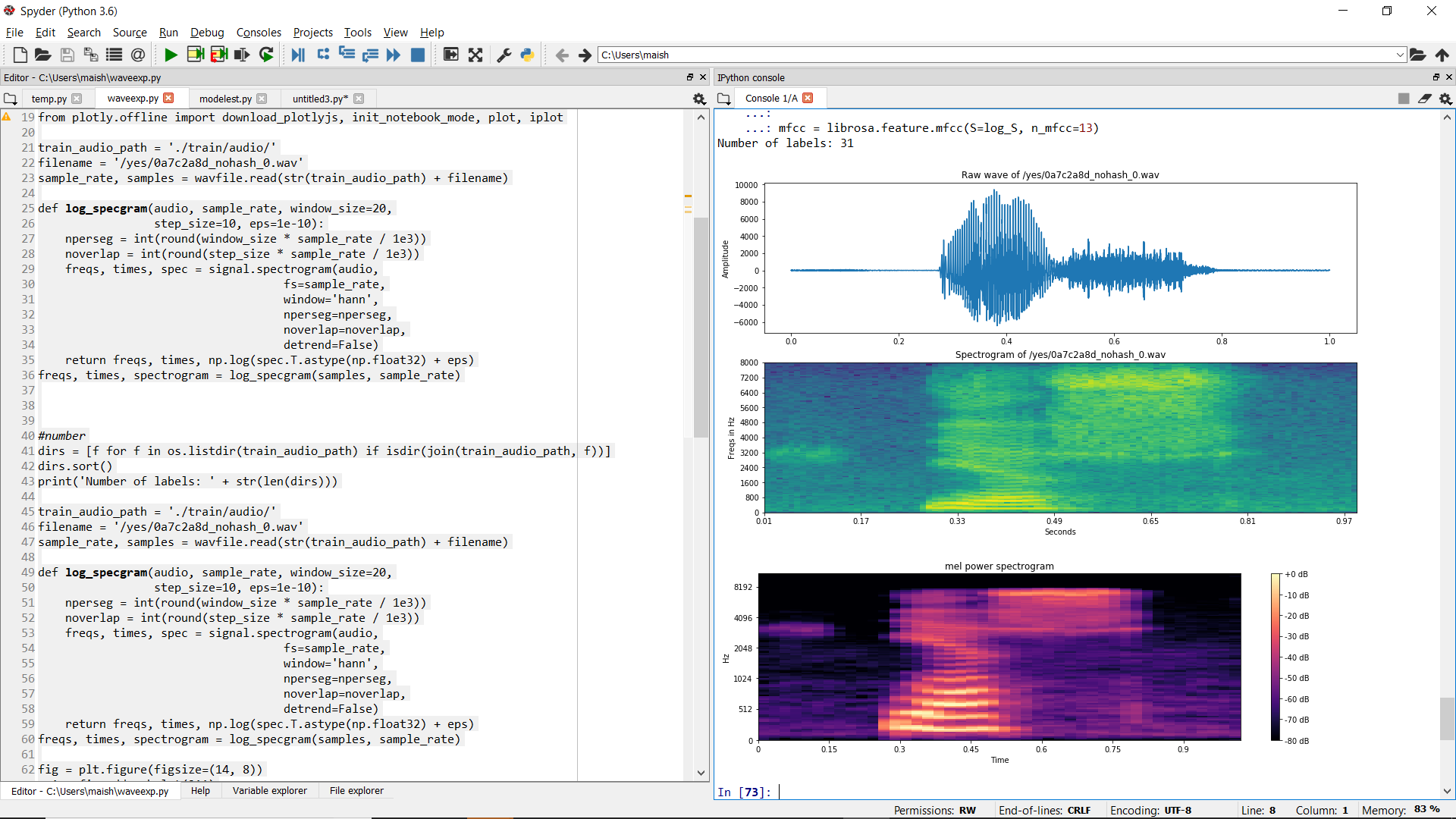
librosa.display.specshow(log\_S, sr=sr, x\_axis='time', y\_axis='mel')

plt.title('mel spectrogram')

plt.colorbar(format='%+02.0f dB')

plt.tight\_layout()

mfcc = librosa.feature.mfcc(S=log\_S, n\_mfcc=13)



## TensorFlow Audio Recognition

Using the speech recognition module released by TensorFlow, we can classify a one second audio clip as either silence, an unknown word, "yes", "no", "up", "down", "left", "right", "on", "off", "stop", or "go”.

Once the environment is setup download the speech command datatset that contains over 65,000 WAVE audio files of people saying thirty different words. This data was collected by Google and released under a CC BY license.

The Zip folder below contains the model script and the dataset. Using this we can run the code on anaconda prompt by using the below command.

python tensorflow/examples/speech\_commands/train.py



Note: Make sure that the command calls the train.py script from the right location.

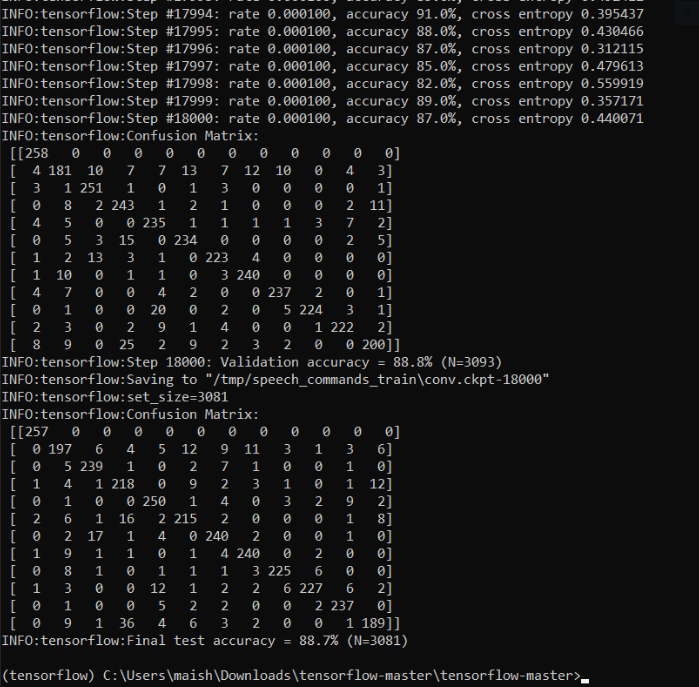
The model works on convolution neural network for small footprint keyword. The audio is converted to images. This is done by grouping the incoming audio samples into short segments, just a few milliseconds long, and calculating the strength of the frequencies across a set of bands. Each set of frequency strengths from a segment is treated as a vector of numbers, and those vectors are arranged in time order to form a two-dimensional array. This array of values can then be treated like a single-channel image, and is known as a spectrogram. Because of TensorFlow's memory order, time in this image is increasing from top to bottom, with frequencies going from left to right, unlike the usual convention for spectrograms where time is left to right. You should be able to see a couple of distinct parts, with the first syllable "Ha" distinct from "ppy".

Because the human ear is more sensitive to some frequencies than others, it's been traditional in speech recognition to do further processing to this representation to turn it into a set of Mel-Frequency Cepstral Coefficients, or MFCCs for short. This is also a two-dimensional, one-channel representation so it can be treated like an image too. If you're targeting general sounds rather than speech you may find you can skip this step and operate directly on the spectrograms.

The image that's produced by these processing steps is then fed into a multi-layer convolutional neural network, with a fully-connected layer followed by a SoftMax at the end.

The accuracy with this model is 88.7%

**Output:**

.