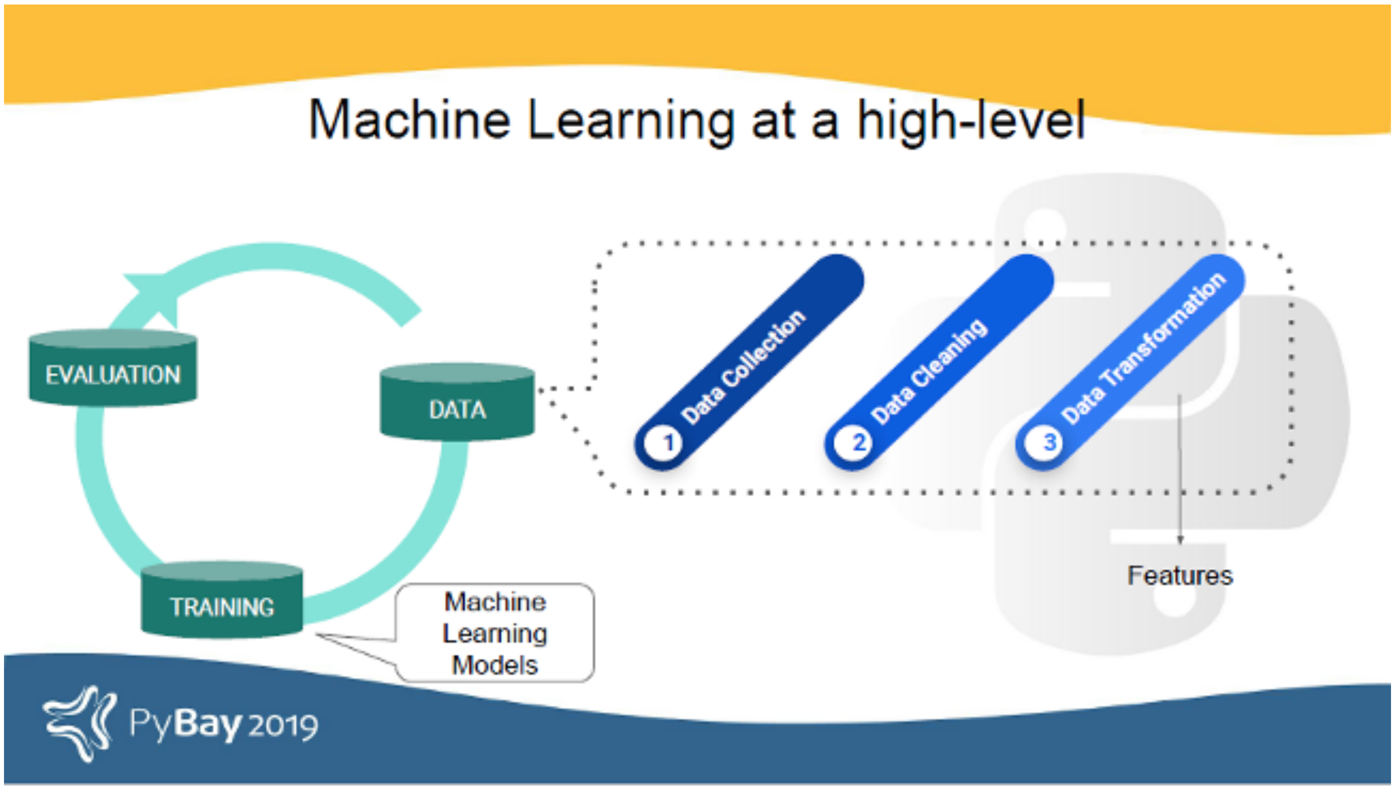
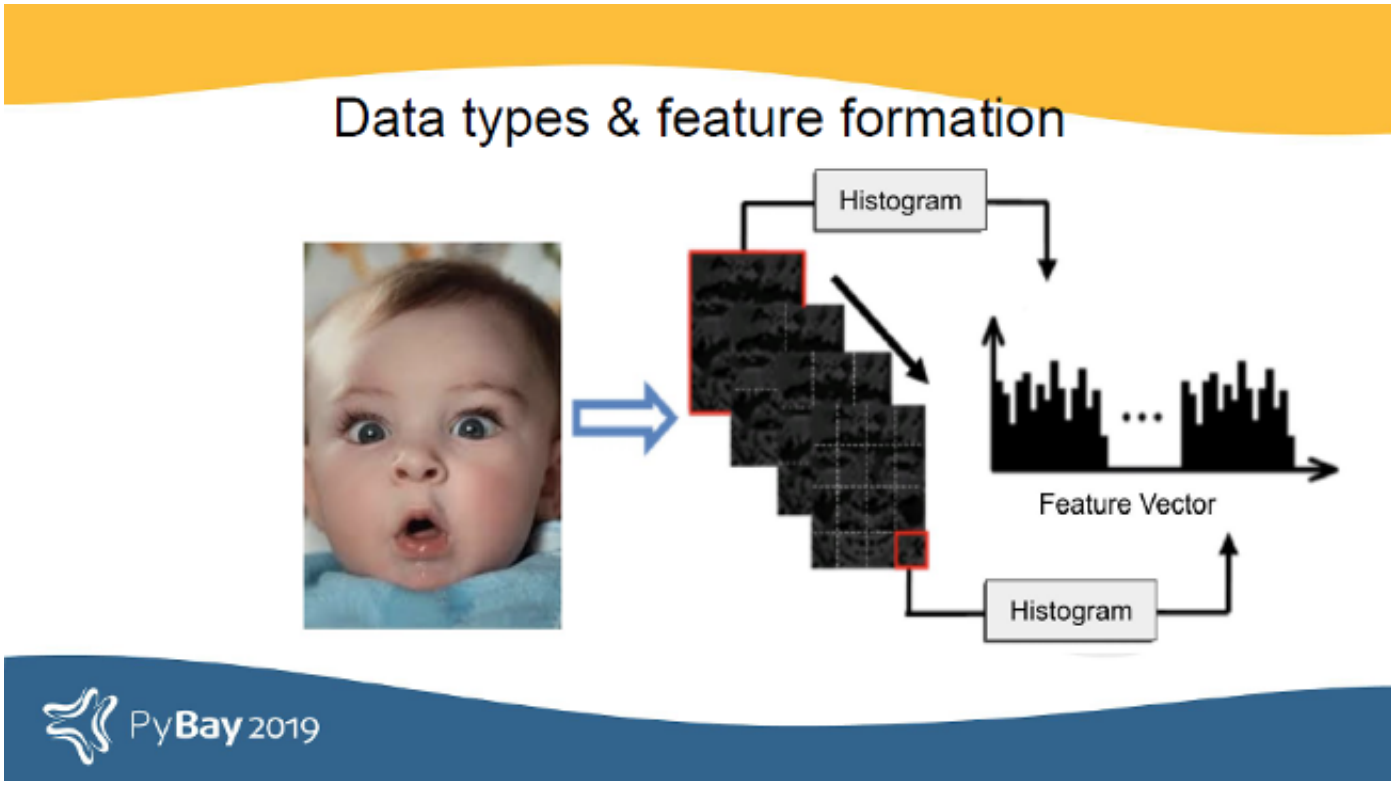
An introduction to audio processing and machine learning using Python

The pyAudioProcessing library classifies audio into different categories and genres.

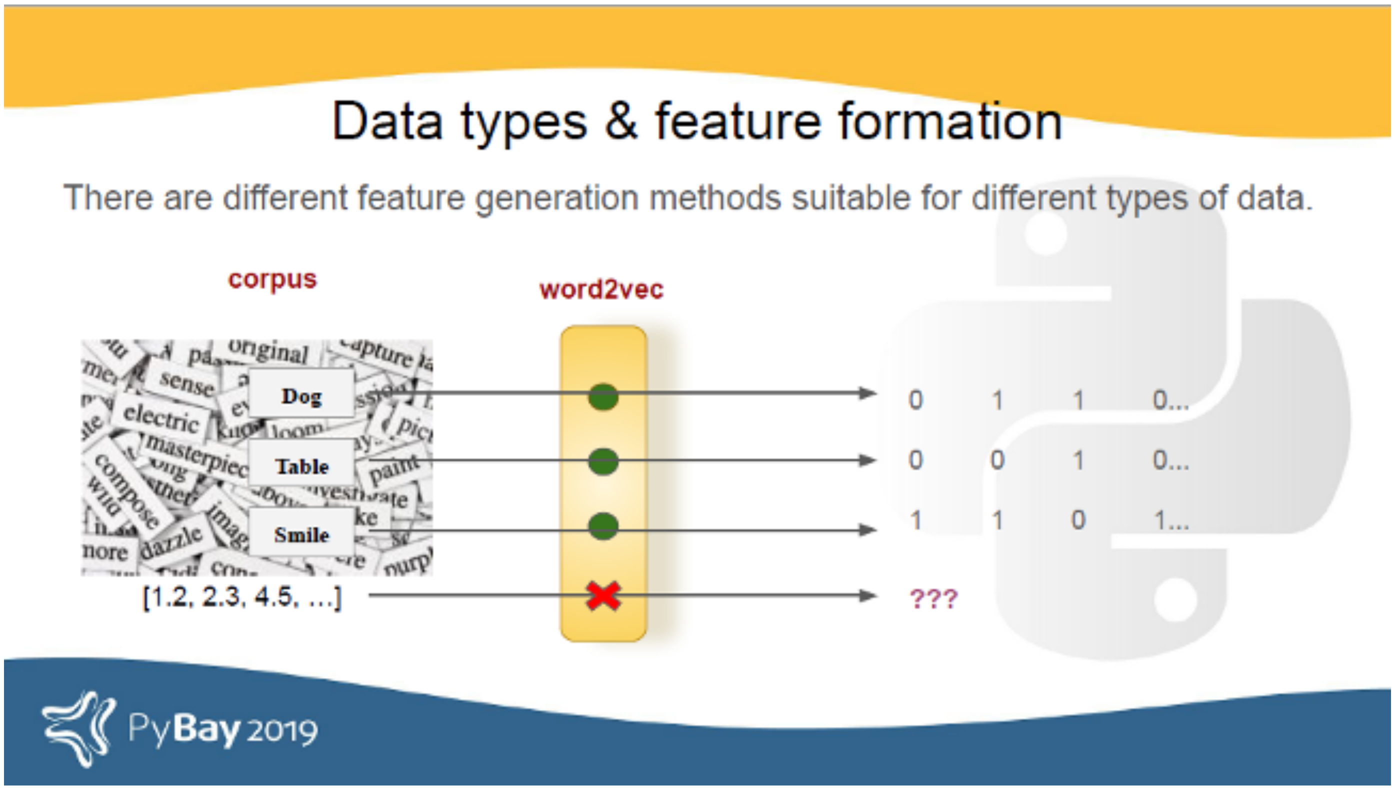
At a high level, any machine learning problem can be divided into three types of tasks: data tasks (data collection, data cleaning, and feature formation), training (building machine learning models using data features), and evaluation (assessing the model). Features, defined as "individual measurable propert[ies] or characteristic[s] of a phenomenon being observed," are very useful because they help a machine understand the data and classify it into categories or predict a value.



Different data types use very different processing techniques. Take the example of an image as a data type: it looks like one thing to the human eye, but a machine sees it differently after it is transformed into numerical features derived from the image's pixel values using different filters (depending on the application).



Word2vec works great for processing bodies of text. It represents words as vectors of numbers, and the distance between two word vectors determines how similar the words are. If we try to apply Word2vec to numerical data, the results probably will not make sense.



Data types & feature formation

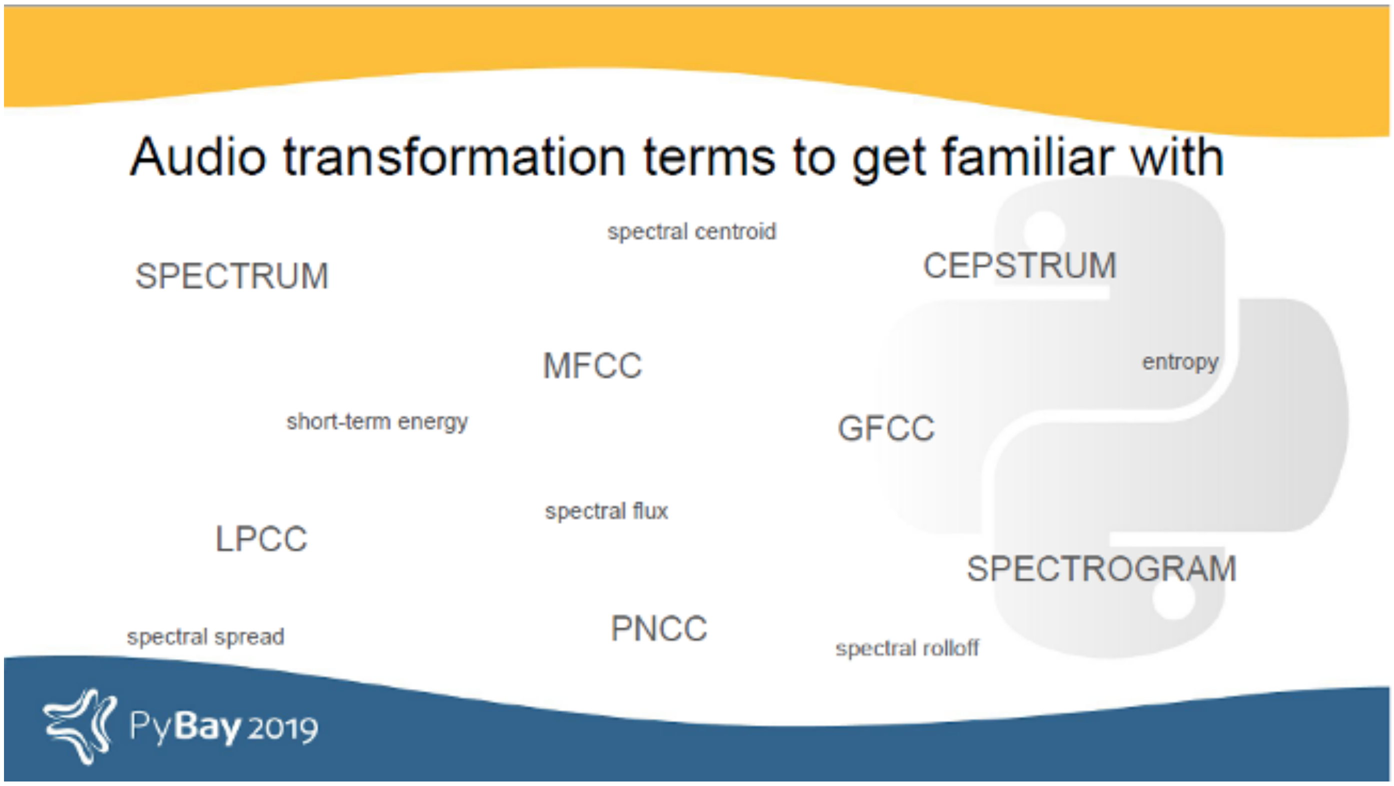
There are different feature generation methods suitable for different types of data.

So, there are processing techniques specific to the audio data type that works well with audio.

What are audio signals?

Audio signals are signals that vibrate in the audible frequency range. When someone talks, it generates air pressure signals; the ear takes in these air pressure differences and communicates with the brain. That's how the brain helps a person recognize that the signal is speech and understand what someone is saying.

There are a lot of MATLAB tools to perform audio processing, but not as many exist in Python. Before we get into some of the tools that can be used to process audio signals in Python, let's examine some of the features of audio that apply to audio processing and machine learning.


Audio transformation terms to get familiar with

SPECTRUM

spectral centroid

CEPSTRUM

MFCC

entropy

short-term energy

GFCC

spectral flux

LPCC

SPECTROGRAM

spectral spread

PNCC

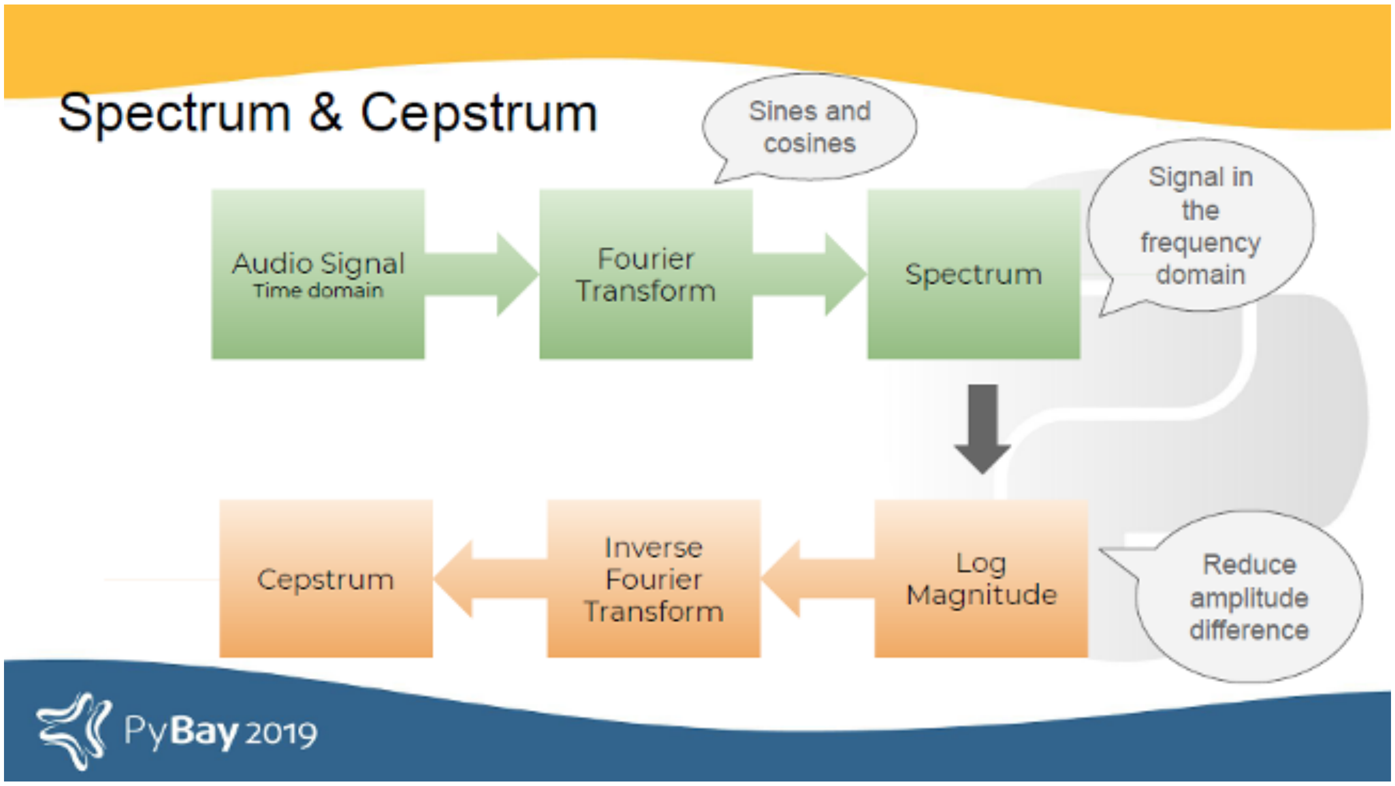
spectral rolloff

Some data features and transformations that are important in speech and audio processing are Mel-frequency cepstral coefficients (MFCCs), Gammatone-frequency cepstral coefficients (GFCCs), Linear-prediction cepstral coefficients (LFCCs), Bark-frequency cepstral coefficients (BFCCs), Power-normalized cepstral coefficients (PNCCs), spectrum, cepstrum, spectrogram, and more.

We can use some of these features directly and extract features from some others, like spectrum, to train a machine learning model.

What are spectrum and cepstrum?

Spectrum and cepstrum are two particularly important features in audio processing.



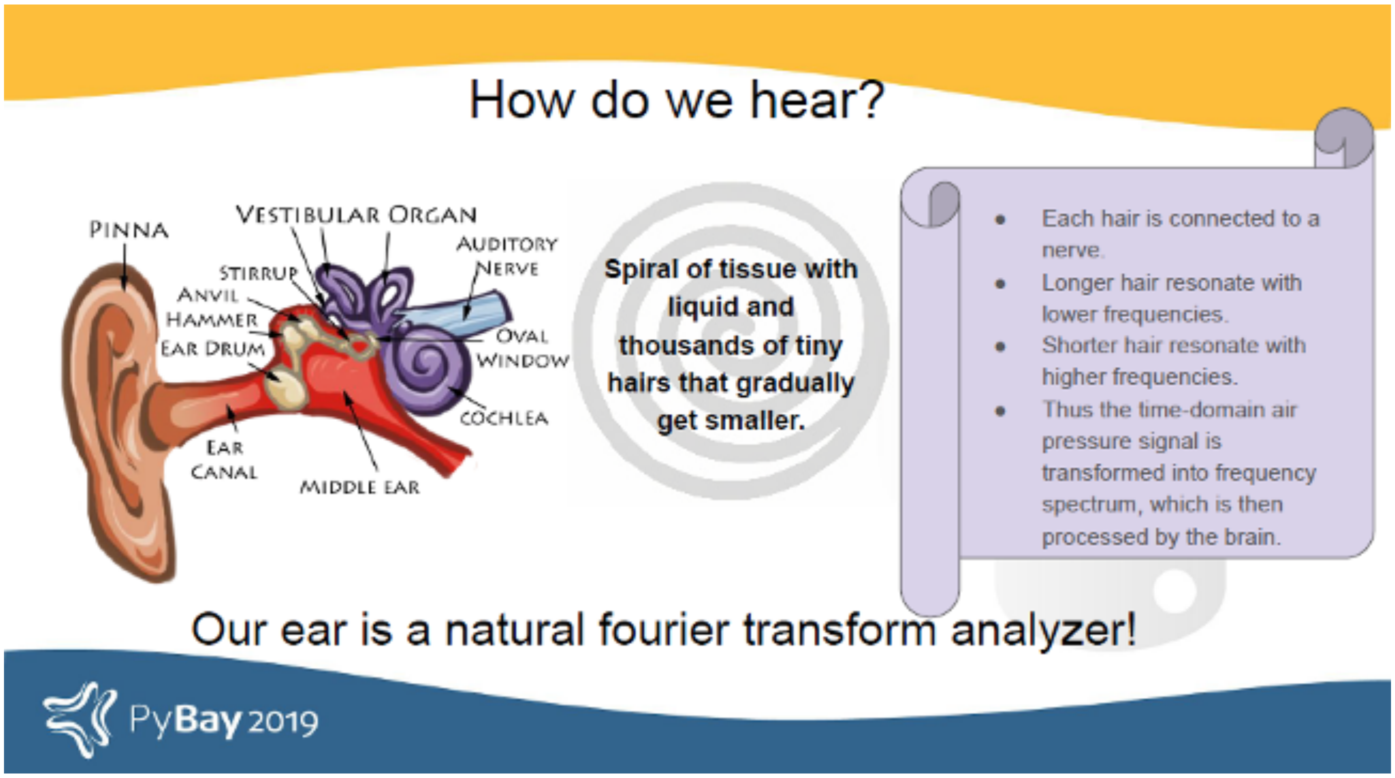
Mathematically, a spectrum is the Fourier transform of a signal. A Fourier transform converts a time-domain signal to the frequency domain. In other words, a spectrum is the frequency domain representation of the input audio's time-domain signal.

A cepstrum is formed by taking the log magnitude of the spectrum followed by an inverse Fourier transform. This results in a signal that's neither in the frequency domain (because we took an inverse Fourier transform) nor in the time domain (because we took the log magnitude prior to the inverse Fourier transform). The domain of the resulting signal is called the quefrency.

What does this have to do with hearing?

More Python Resources

The reason we care about the signal in the frequency domain relates to the biology of the ear. Many things must happen before we can process and interpret a sound. One happens in the cochlea, a fluid-filled part of the ear with thousands of tiny hairs that are connected to nerves. Some of the hairs are short, and some are relatively longer. The shorter hairs resonate with higher sound frequencies, and the longer hairs resonate with lower sound frequencies. Therefore, the ear is like a natural Fourier transform analyzer!


PINNA
VESTIBULAR ORGAN
AUDITORY NERVE
STIRRUP
ANVIL
HAMMER
EAR DRUM
OVAL WINDOW
COCHLEA
EAR CANAL
MIDDLE EAR
PyBay 2019

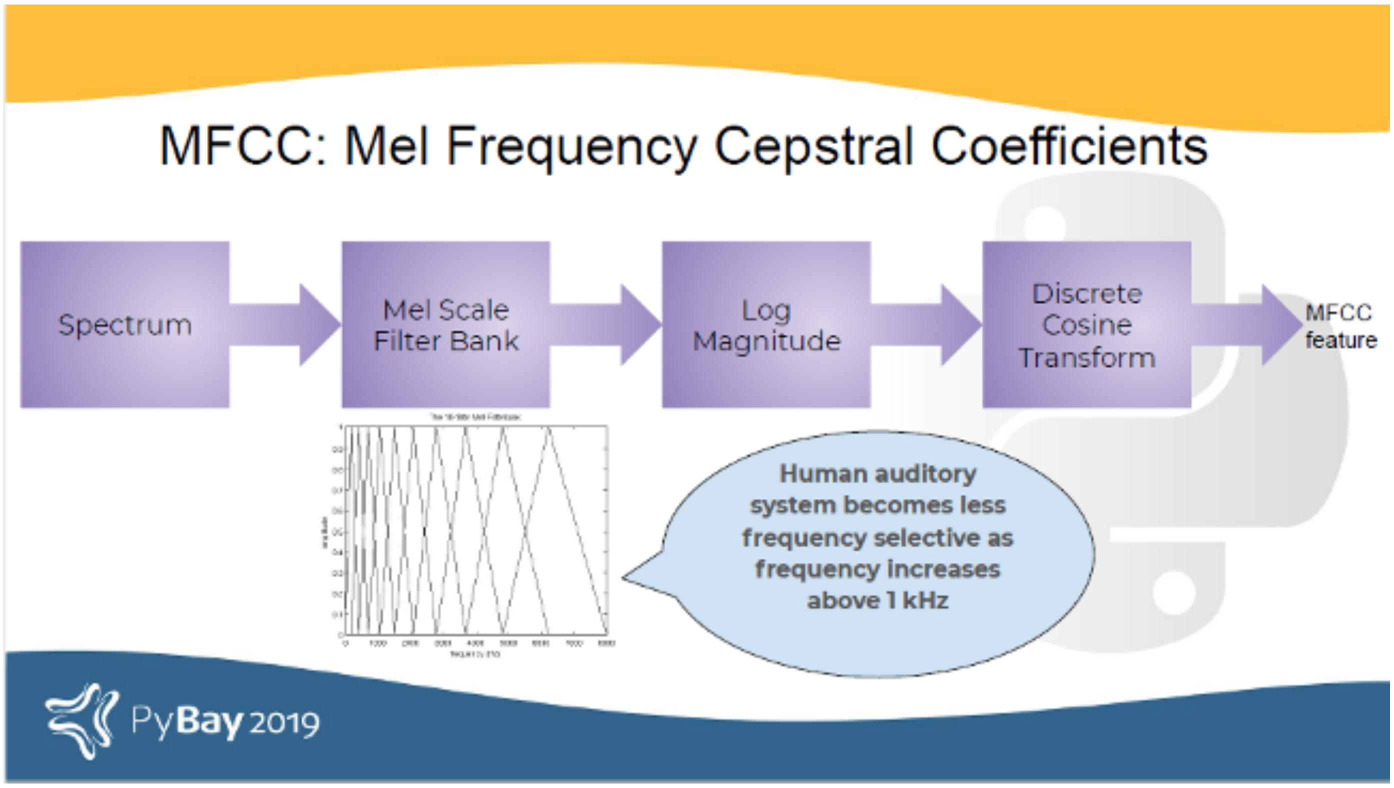
How do we hear?

Spiral of tissue with liquid and thousands of tiny hairs that gradually get smaller.

- Each hair is connected to a nerve.
- Longer hair resonate with lower frequencies.
- Shorter hair resonate with higher frequencies.
- Thus the time-domain air pressure signal is transformed into frequency spectrum, which is then processed by the brain.

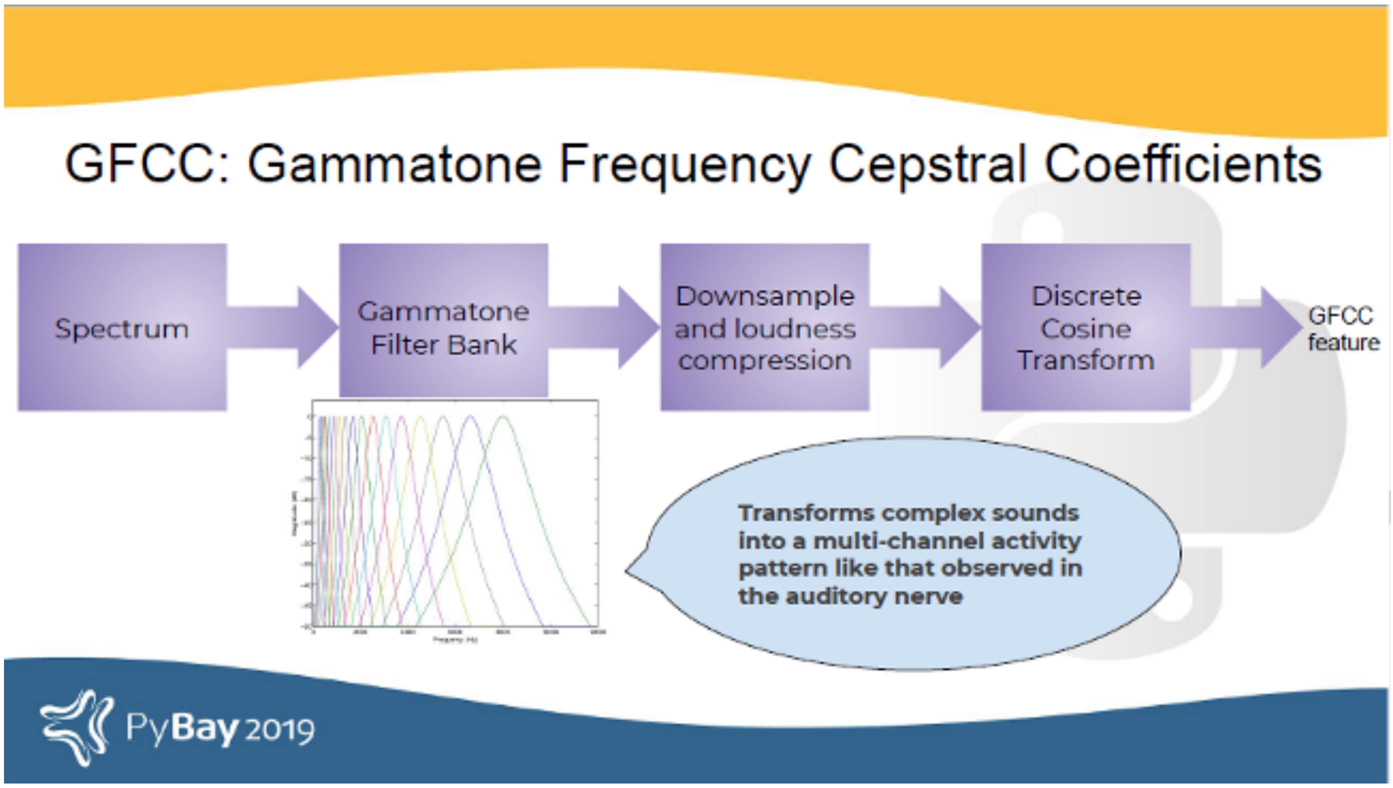
Our ear is a natural fourier transform analyzer!

Another fact about human hearing is that as the sound frequency increases above 1kHz, our ears begin to get less selective to frequencies. This corresponds well with something called the Mel filter bank.


MFCC: Mel Frequency Cepstral Coefficients
Spectrum
Mel Scale Filter Bank
Log Magnitude
Discrete Cosine Transform
MFCC feature
Human auditory system becomes less frequency selective as frequency increases above 1 kHz
PyBay 2019

Passing a spectrum through the Mel filter bank, followed by taking the log magnitude and a discrete cosine transform (DCT) produces the Mel cepstrum. DCT extracts the signal's main information and peaks. It is also widely used in JPEG and MPEG compressions. The peaks are the gist of the audio information. Typically, the first 13 coefficients extracted from the Mel cepstrum are called the MFCCs. These hold very useful information about audio and are often used to train machine learning models.

Another filter inspired by human hearing is the Gammatone filter bank. This filter bank is used as a front-end simulation of the cochlea. Thus, it has many applications in speech processing because it aims to replicate how we hear.



GFCC: Gammatone Frequency Cepstral Coefficients

Spectrum
Gammatone Filter Bank
Downsample and loudness compression
Discrete Cosine Transform
GFCC feature
Transforms complex sounds into a multi-channel activity pattern like that observed in the auditory nerve
PyBay 2019

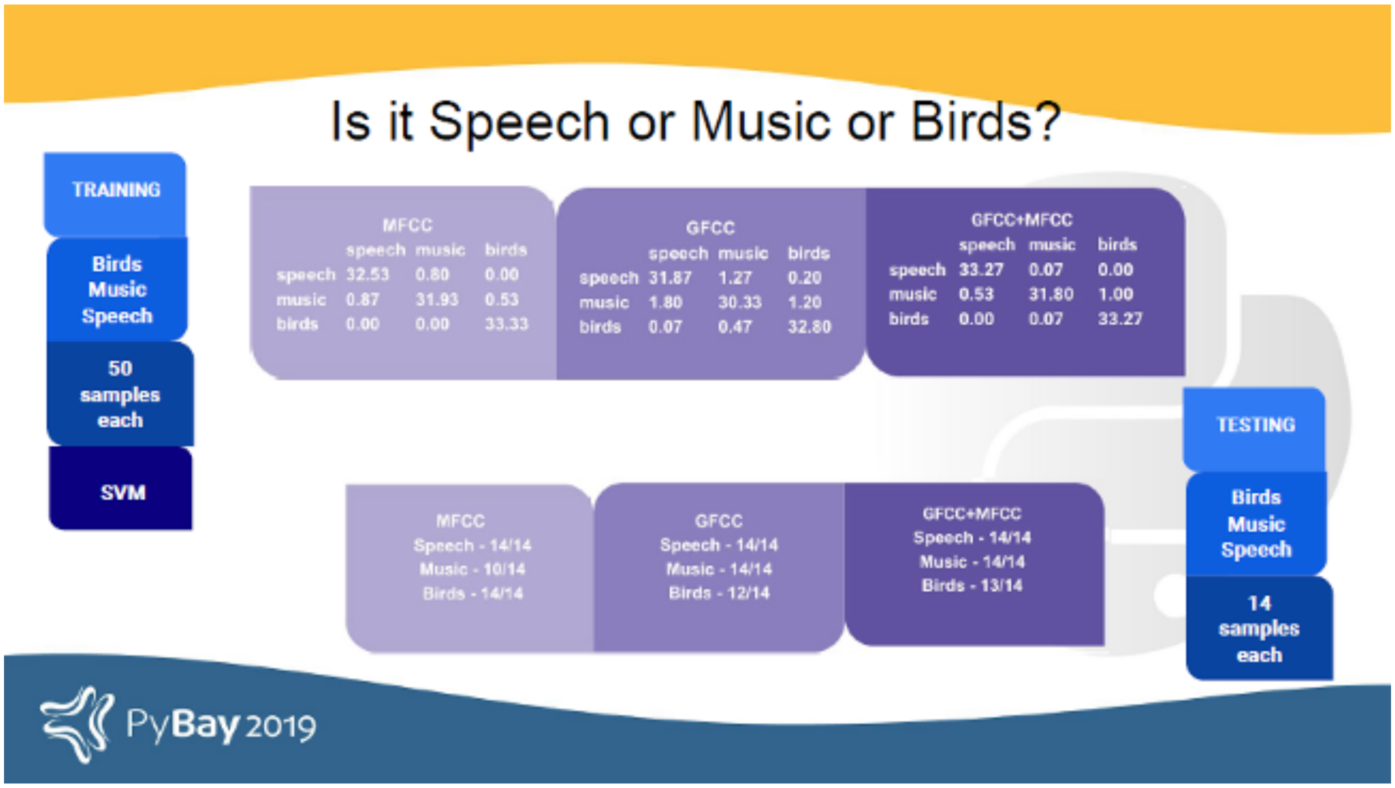
GFCCs are formed by passing the spectrum through Gammatone filter bank, followed by loudness compression and DCT. The first (approximately) 22 features are called GFCCs. GFCCs have a number of applications in speech processing, such as speaker identification.

Other features useful in audio processing tasks (especially speech) include LPCC, BFCC, PNCC, and spectral features like spectral flux, entropy, roll off, centroid, spread, and energy entropy.

Building a classifier

As a quick experiment, let's try building a classifier with spectral features and MFCC, GFCC, and a combination of MFCCs and GFCCs using an open source Python-based library called pyAudioProcessing.

To start, we want pyAudioProcessing to classify audio into three categories: speech, music, or birds.



Is it Speech or Music or Birds?

TRAINING

Birds
Music
Speech

50 samples each

SVM

MFCC

	speech	music	birds
speech	32.53	0.80	0.00
music	0.87	31.93	0.53
birds	0.00	0.00	33.33

GFCC

	speech	music	birds
speech	31.87	1.27	0.20
music	1.80	30.33	1.20
birds	0.07	0.47	32.80

GFCC+MFCC

	speech	music	birds
speech	33.27	0.07	0.00
music	0.53	31.80	1.00
birds	0.00	0.07	33.27

TESTING

Birds
Music
Speech

14 samples each

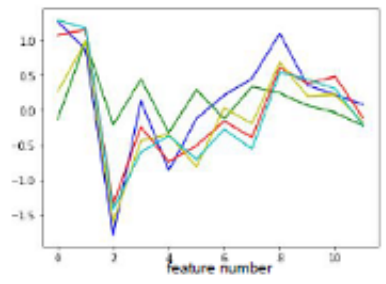
MFCC
Speech - 14/14
Music - 10/14
Birds - 14/14

GFCC
Speech - 14/14
Music - 14/14
Birds - 12/14

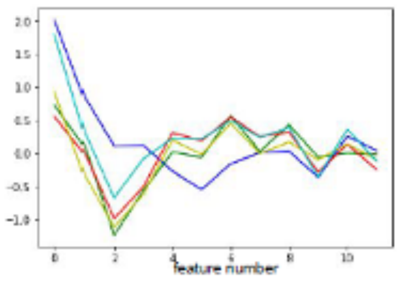
GFCC+MFCC
Speech - 14/14
Music - 14/14
Birds - 13/14

PyBay 2019

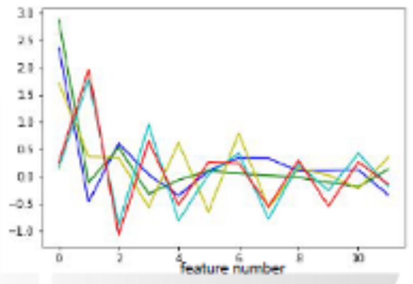
Using a small dataset (50 samples for training per class) and without any fine-tuning, we can gauge the potential of this classification model to identify audio categories.



MFCC of a speech signal



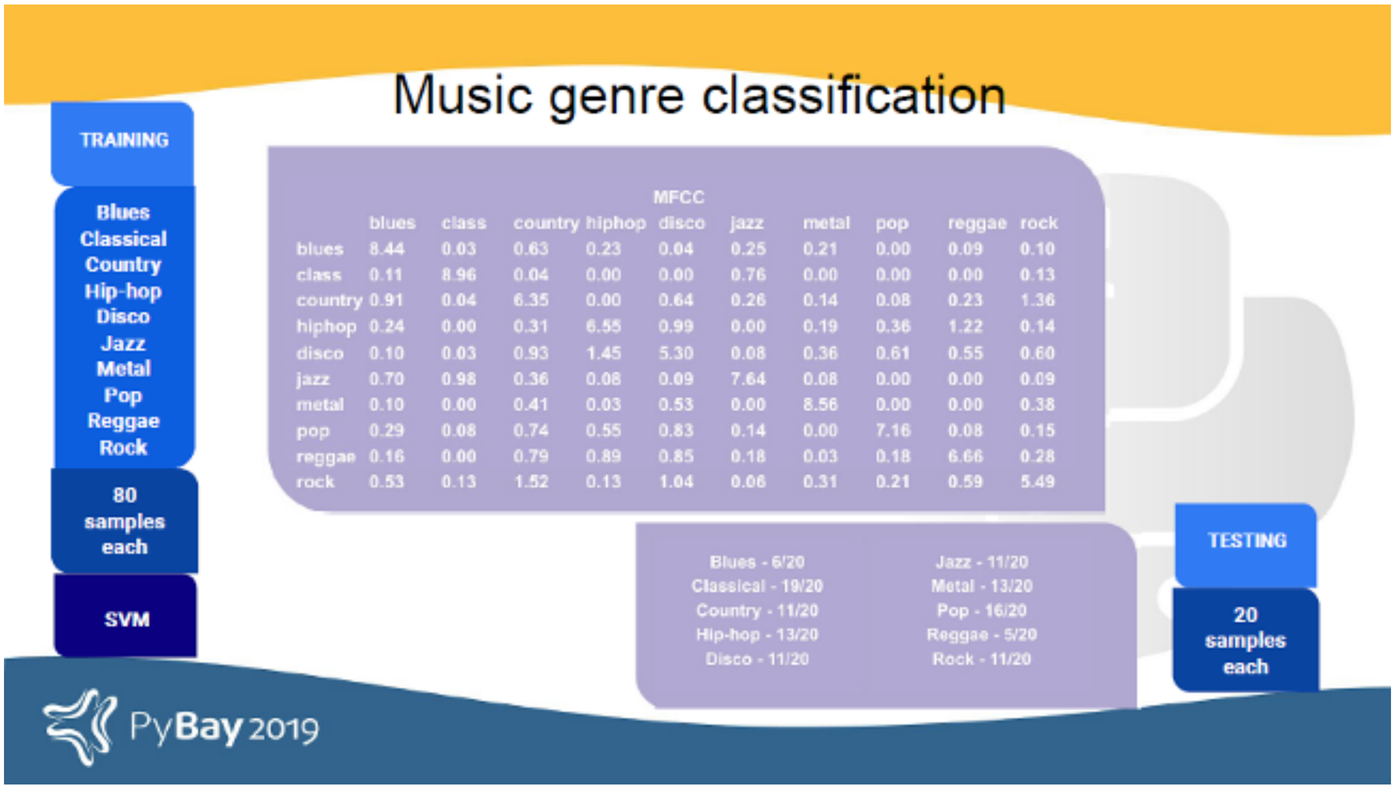
MFCC of a music signal



MFCC of a bird signal

PyBay 2019

Next, let's try pyAudioProcessing on a music genre classification problem using the GZTAN audio dataset and audio features: MFCC and spectral features.



Music genre classification

TRAINING

Blues
Classical
Country
Hip-hop
Disco
Jazz
Metal
Pop
Reggae
Rock

80 samples each

SVM

MFCC

	blues	class	country	hiphop	disco	jazz	metal	pop	reggae	rock
blues	8.44	0.03	0.63	0.23	0.04	0.25	0.21	0.00	0.09	0.10
class	0.11	8.96	0.04	0.00	0.00	0.76	0.00	0.00	0.00	0.13
country	0.91	0.04	6.35	0.00	0.64	0.26	0.14	0.08	0.23	1.36
hiphop	0.24	0.00	0.31	6.55	0.99	0.00	0.19	0.36	1.22	0.14
disco	0.10	0.03	0.93	1.45	5.30	0.08	0.36	0.61	0.55	0.60
jazz	0.70	0.98	0.36	0.08	0.09	7.64	0.08	0.00	0.00	0.09
metal	0.10	0.00	0.41	0.03	0.53	0.00	8.56	0.00	0.00	0.38
pop	0.29	0.08	0.74	0.55	0.83	0.14	0.00	7.16	0.08	0.15
reggae	0.16	0.00	0.79	0.89	0.85	0.18	0.03	0.18	6.66	0.28
rock	0.53	0.13	1.52	0.13	1.04	0.06	0.31	0.21	0.59	5.49

Blues - 6/20	Jazz - 11/20
Classical - 19/20	Metal - 13/20
Country - 11/20	Pop - 16/20
Hip-hop - 13/20	Reggae - 5/20
Disco - 11/20	Rock - 11/20

TESTING

20 samples each

PyBay 2019

Some genres do well while others have room for improvement. Some things that can be explored from this data include:

- Data quality check: Is more data needed?
- Features around the beat and other aspects of music audio
- Features other than audio, like transcription and text
- Would a different classifier be better? There has been research on using neural networks to classify music genres.

Regardless of the results of this quick test, it is evident that these features get useful information out of the signal, a machine can work with them, and they form a good baseline to work with.

Learn more

Here are some useful resources that can help in your journey with Python audio processing and machine learning:

- pyAudioAnalysis
- pyAudioProcessing
- Power-normalized cepstral coefficients (PNCC) for robust speech recognition
- LPCC features
- Speech recognition using MFCC
- Speech/music classification using block-based MFCC features
- Musical genre classification of audio signals
- Libraries for reading audio in Python: SciPy, pydub, libROSA, pyAudioAnalysis
- Libraries for getting features: libROSA, pyAudioAnalysis (for MFCC); pyAudioProcessing (for MFCC and GFCC)
- Basic machine learning models to use on audio: sklearn, hmmlearn, pyAudioAnalysis, pyAudioProcessing

This article is based on Jyotika Singh's presentation "Audio processing and ML using Python" from PyBay 2019.