2018

Information System & Machine Learning Lab

University of Hildesheim, Germany

Emotion Detection from Speech

Student Research Project Report

Niraj Dev Pandey | MSc. Data Analytics

SUBMITTED TO | Prof. Dr. Dr. Lars Schmidt-Thieme

Table of Contents

1. Abstract

Talked dialect passes on two types of information: value-based or transactional (content, what is said) and interactional (how it is said). The value-based or transactional message shared during any communication has been studied and researched pretty much in different circumstances and in people of different ages using many tests benchmarks of speech intelligibility. However, the other part of the speech information, interactional one has been more limited and need to be discovered extensively [1]. Extracting the emotional state of a speaker from his or her semantic speech uttered in different circumstances is the objective of this research. It has the potential to change the way human interact with intelligent machines. This is the era of AI and machine learning. In near future, emotion detection from speech could help simulate more realistic avatar interaction.

In this project, simulated English emotional speech database has been borrowed from a Toronto Speech Data Set 2010 which was collected by University of Toronto. This data-set has 7 emotion classes (anger, disgust, fear, happiness, pleasant surprise, sadness, neutral). Emotional classification is attempted on the corpus using spectral features. Our spectral features include MFCC, Mel, Tonnetz, Croma & Contrast. 193 were the total size of feature vector. These features give us prosody in a particular speech. Where prosody means, the patterns of rhythm and sound used in speech or the patterns of stress and intonation in a language. Basically, it is knowing the energy present in the speech.

This dataset is trained on multiple classifiers, wherein with each classifier, related learning and error penalty rate parameters have been varied to find the best set of classifiers which fits to the data perfectly. The lists of accuracies and loss are compared. Our methods show that Extra Tree Classifier provides the best accuracy rates on test data-set, with accuracy being 99.82 and neural network also showed very good performance having 99.64 percentage of accuracy score. During the research project many other classification approaches were tried which leads us to compare them with each other.

2. Motivation

The emotion detection from speech was motivated for using this technology into business. Emotion detection from speech can be applied at various places to allocate limited human resources to clients with the highest levels of requirements, such as in automated call centers or in any other places where we want to make things automated with less human forces. These kinds of technologies will revolutionize many industries where we can reduce the work-load on human employees. My motivation to work on this project was to make human-machine interaction not only command based but rather having emotion detail of that interaction. Which will lead us to the point where we can make more interactive avatar of machines. I personally like to talk with machines and played numerous times with Apple's Siri and Microsoft's Cortana. These personal assistant systems would be very great when incorporated with speech's emotion understanding. I can imagine the time when machines will start understanding our command along with emotions very well, that moment will be revolutionary in the field of technology.

Let's think for the sake of our understanding the example of automated call centers where all the calls would be automated i.e. first you will ask your queries to a machine and if it would be unable to answer and when it will figure out that the person is getting annoyed or impatient with the answer provided it will redirect the client to a human attender. Another application could be sorting voice mail messages as per the emotions expressed by the clients/customer's call (also see the figure 01). There are many research out there which shows the application of emotion detection in depth. Such as the one mentioned in the reference section [3]. Another motivation can be to use such system in hospitals or clinics as claimed by a researcher Alex and team [4] in 2011 where they said, and I quote

"While a human (in clinic case) may not be available to assist everyone, automated emotion detection can be used to "triage" a customer and, if they are growing angry or impatient, transfer them to speak with a human. It may also be useful for helping autistic children and adults learn to recognize more subtle social cues, or help people re ne their speaking skills so that they have better control over the message they send."

3. Introduction

Understanding human-speech and the emotion it holds has been a vital and interesting part of AI and machine learning research for quite a while. Emotion detection systems could be used to improve services (see fig below) by being adaptive to the user's emotions along with their commands.

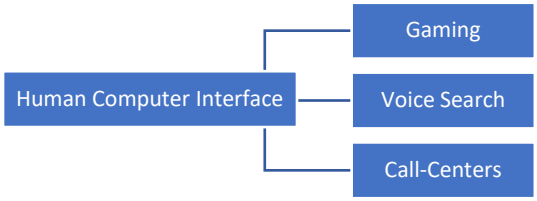


Fig. 01. It shows the area where such technology can be deployed and benefit the

Same expressions/words can pass on different emotion when uttered in a different way. The research in the field of emotion detection is not limited anymore as it was earlier. At present, we have already discovered the most influential features that dominates the emotion to be in that particular class. Whereas, when we deal with many emotion classes then there is room for miss-classification. Meaning, there are some emotions which holds similar feature statistics and leads to miss-classification.

Here, we have 7 classes meaning seven emotions in the data-set borrowed from University of Toronto. The data was first used by a PhD. student named Katherine Lise Dupuis for her PhD. thesis. Reference [1] is her thesis submission where one can look for more detail.

In this research, we endeavor to address these issues by extracting features that were never extracted before for emotion detection from speech (from best of my knowledge). We utilize K-Means, Decision Tree, Extra Tree Classifier and most beloved model Neural Network to classify opposing emotions. We have extracted many of temporal and spectral features that can be obtained from human voice. To extract feature the python library called Librosa was used. Then we will use data relating to our feature which is croma, tonnetz, contrast & Mel Frequency Cepstral Coefficients (MFCCs) as inputs feature vector to classification algorithms.

4. State-of-the-art

The emotion detection from speech is not a novel research area anymore. But still many researchers are trying to find more accurate way to extracting speech features. So far Praat and other feature extraction software were in use. Later people shifted to python library called python_feature_extraction. Neural Networks have been successfully used for speech analysis, speech emotion recognition, and natural language processing [14]. Before 1980 SSC was the prime feature in the process of emotion detection from speech, but after 80's MFCC took over the game and till today it dominates the emotion or emotional feature in a given voice/speech. In the past years there were many research debating about the acted and real emotion. The way you collect the data have been prime task among researchers. How much acted emotional speech translate to the real/genuine emotion [15]. The svm classifier was most beloved amongst signal processing before neural networks. So far from best of my knowledge researchers have achieved approximately 85-90% of accuracy while detecting emotion form speech. We have already witnessed speech emotion detection by using old speaker while training to detect young speaker emotion speech while testing the model. There is some good work on "speech emotion recognition by younger and older adults and effects on intelligibility" [1].

5. Our Approach

Since there are several researches out there which has achieved very good results and we are going to take privilege of those old works and will enhance the prediction power along with having more features that haven't used before. We are going to used Librosa as a feature detection from speech not conventional way (Praat, python_feature_extraction). We are going to extract not just MFCC but Chroma, Tonnetz & LPCC too. For the classification of emotions, we have used most popular algorithm Neural Networks. From best of my knowledge, we have used AutoML for the first time to get optimal parameters along with classifier name. We will get to see TPOTs AutoML in the experiment section. Our approach consists of both old and young speaker mixed equally and still showed very good performance in predicting correct emotion class.

6. Corpus of Emotional Speech Data

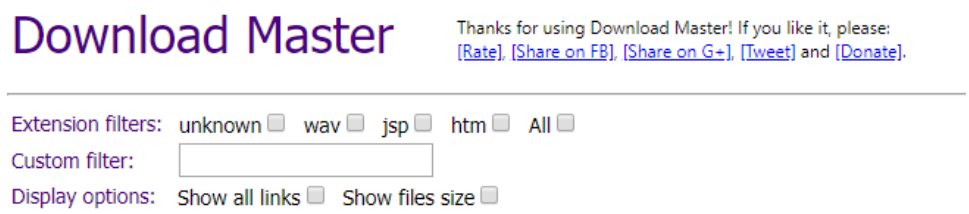
The data-set is publicly available for research purposes. It has a interesting name too “Toronto emotional speech set (TESS) - Older talker_Happy”. Here is the web-site address [4]. A set of 200 target words were spoken in the carrier phrase "Say the word __xyz___" by two actresses (aged 26 and 64 years) and recordings were made of the set portraying each of seven emotions (anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral). The age of the actresses gives us freedom to extend this research to find other interesting facts within this domain. There are 2800 stimuli in total. The two actresses were recruited from the Toronto area. Both actresses speak English as their first language, are university educated, and have musical training. Audiometric testing indicated that both actresses have thresholds within the normal range. The above data-set description is from the web-site itself [4]. The author of the data-set were Dupuis Kate, Pichora-Fuller & M. Kathleen. The data of issuing this corpus was 21-Jun-2010.

Emotion Class	Number of Sample
Anger	400
Disgust	400
Fear	400
Happiness	400
Pleasant Surprise	400
Sadness	400
Neutral	400
∑ = 7 Classes	∑ = 2800 Stimuli

table 01. It shows the basic statistics about the data-set. Here one can see that we have 7 emotions and every emotion class has 400 samples. One thing we need to notice here that, there are two speakers young and old. Therefore, from 1 class 200 samples were spoken by young speaker and another 200 by old speaker that counts 400 samples for every class.

The corpus is not available as whole in one folder or click and get. therefore, it has to be downloaded manually file by file. It means that, one has to click 2800x2 = 5,600 times. Meaning I spent too much time to download whole 2800 samples. Later I found some quick way to download all .wav files in few mouse clicks.

There are some web browser extensions which will help us to download all the .wav files present in the respective web page. For google Chrome user you can go for Download Master extension here (https://goo.gl/iyCBia). This extension has many options that enable us to download files quickly.



Open the data-set website and you just need to click on the extension and select wav format from extension filters and tap on download.

6.1 Corpus Division Strategy

Data-set division strategy has one of the big impact on our final prediction. Therefore, this is crucial part to decide how much data you want to invest in training and how much for testing. Most machine leaning research follows traditional way to dividing the corpus. Similarly, in this research conventional way of data-division strategy were used. Meaning, the training and test data was divided with the ratio of 80-20%. The table below shows how much data were used to train and test from each emotion class. We must use same number of samples from each emotion class to train and then to test. That's the way to check how accurate your model is performing.

Labels	Train/Test Samples
Anger	320/80
Disgust	320/80
.	.
.	.
Neutral	320/80
	∑ = 2,240/560

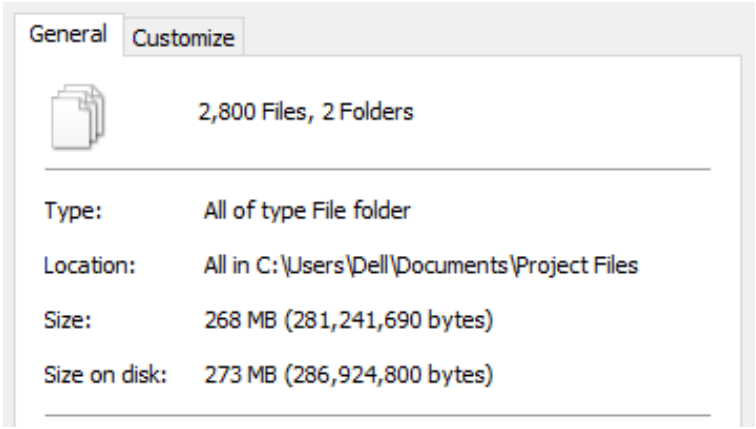
The table 02 shows the statistics about the data-set division strategy. Here you already know that we have 7 emotions and every emotion class has 400 samples. From that 400 320 samples will be used to train the model and remaining 80 samples will be utilized to test the model performance.

So, altogether we have 2240 samples to train the model and 560 samples to test the model accuracy.

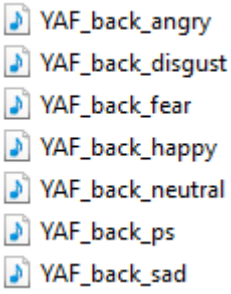
6.2 How corpus look like ?

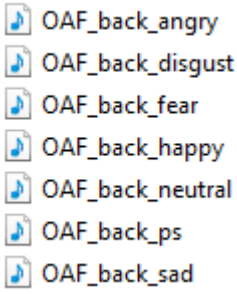
The corpus is not really big therefore, it is easy to train in short time. Every sample contain just one-word utterance in all seven circumstances (emotion classes). Here is some example.

Here, one can see the size of the data-set. Here, 2 folder means train and test folder which counts to 2800 stimuli together. The size of the corpus in mega-bytes counts to 268 MB. The files or samples are in .wav format. Apart from this the file name itself are pretty explanatory.



Here one can see the file naming format. I contain a lot of information itself that gives us pretty good idea that what has spoken and who spoke that word. See the name table below.

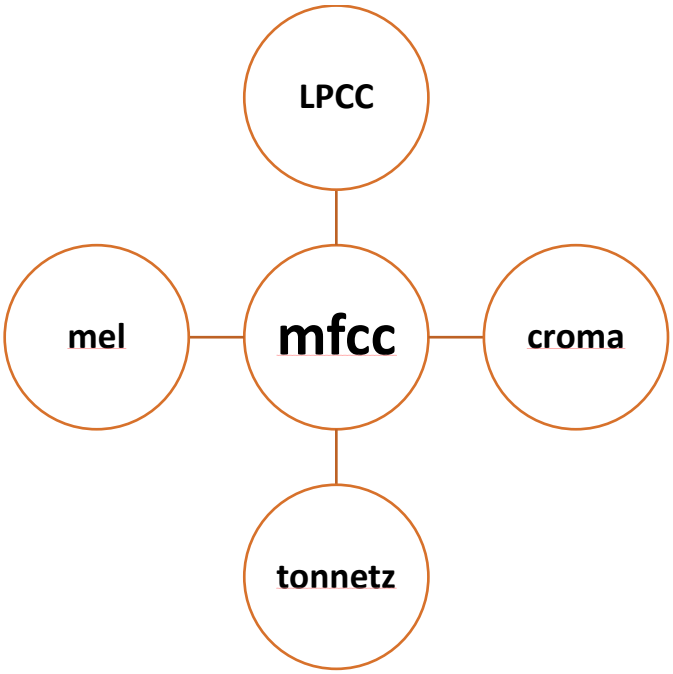




Here YAF_back_angry is the wav file name containing some information. Such as YAF stands for young speaker and word back is the utterance in this speech sample and at last it has the emotion of saying back. Similarly, in the other column we have same word spoken with similar emotion, but this is old actress. OAF stands for old speaker here.

7 Feature Extraction

Speech is one of the oldest ways to express our thoughts and ideas. But today the story is different we can use speech data to recognize biometric and also to use voice lock and other machine human interaction. Every voice or speech has some features and the speech uttered in different circumstances has different characteristics or feature stats. Extraction of features are very crucial in this research project, since these features helps us to make difference between two speeches. The more better feature better the prediction accuracy. For this research we are going to use below features –

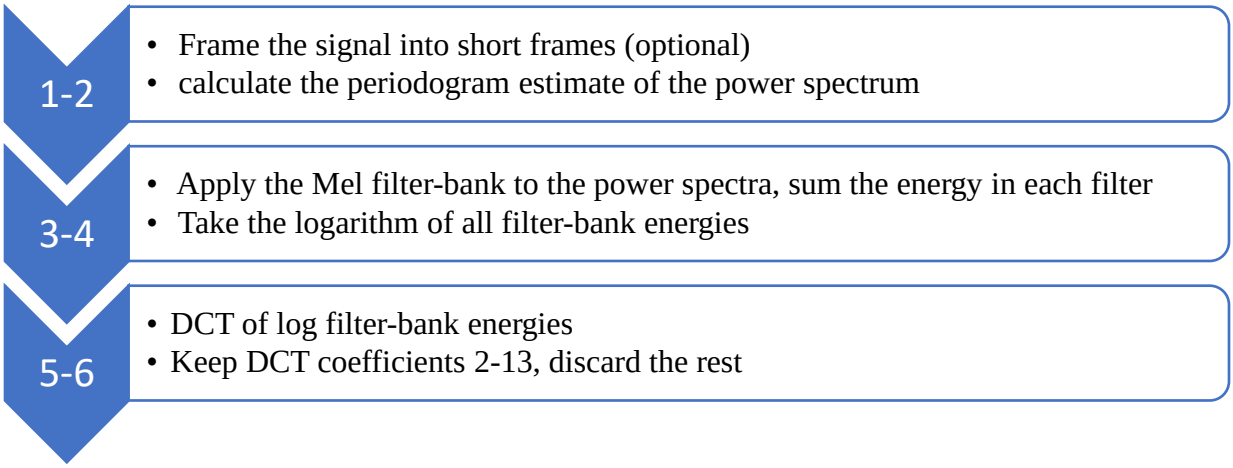


Earlier it was SSC which used to rule the speech feature extraction before 1980. Whereas, after 1980 MFCC (introduce by Davis and Mermelstein) came into the field and overperformed ssc and became dominant in the area of speech processing. In the very beginning waveform used to contain some information or statistics about the voice or speech but when the question is about emotion detection many features doesn't have ability to extract those information's which dominates emotion in any given speech. Reference [2] showed really in-depth research on speech feature extraction. There are several ways to extract speech features. Such as, Prat and other libraries (python speech feature & Librosa) according to programming languages. We are going to use Librosa to achieve feature extraction objective.

7.1 MFCC - Mel-frequency cepstral coefficients

There are many steps involve extracting mfcc from the speech. Let's start from the high-level steps at glace and then we will go in-depth analysis. This section 6 is based on the mfcc tutorial [5] and the research paper [2]. Below are the steps involved in mfcc feature extraction explained very well in reference [5].

Steps at glance -



- Power spectrum – Identify frequency present in that frame
- Mel filter-bank - clumps of periodogram bins and sum them up to get an idea of how much energy exists in various frequency regions
- DCT - decorrelates the energies

Now, the question is why are we performing above steps? What are the ideas and reason behind taking such steps? A speech signal is constantly changing every second or maybe quicker than that, so to simplify things we break entire speech into short frames because on short time scales the audio signal doesn't change much and there we can find some stationary statistics. Therefore, we frame the speech signal into 20-40ms frames.

One should choose this specific frame so that you will get enough samples to achieve reliable spectrum statistics that lead us to a conclusion that we should not choose longer or shorter than 20-40ms frames that would not be useful [5]. Since this step in optional so, this project did not perform this step and got perfect results still. The recommended frame size for mfcc is 25ms.

The formula for converting from frequency to Mel scale borrowed from [5] is:

$$M(f) = 1125 \ln(1 + f/700) \qquad (1)$$

To go from Mel back to frequency:

$$M^{-1}(m) = 700(\exp(m/1125) - 1) \qquad (2)$$

This mfcc is based on human hearing and nature of our dealing with speech so next step is to calculate the power spectrum of each frame. Calculating power spectrogram helps us to know that certain frequencies are present in a given frame. The periodogram estimate performs a similar kind of role, recognizing which frequencies exists in that particular frame. Here is a diagram borrowed from [5]. Which will help us to make better understanding of mfcc and it's step too.

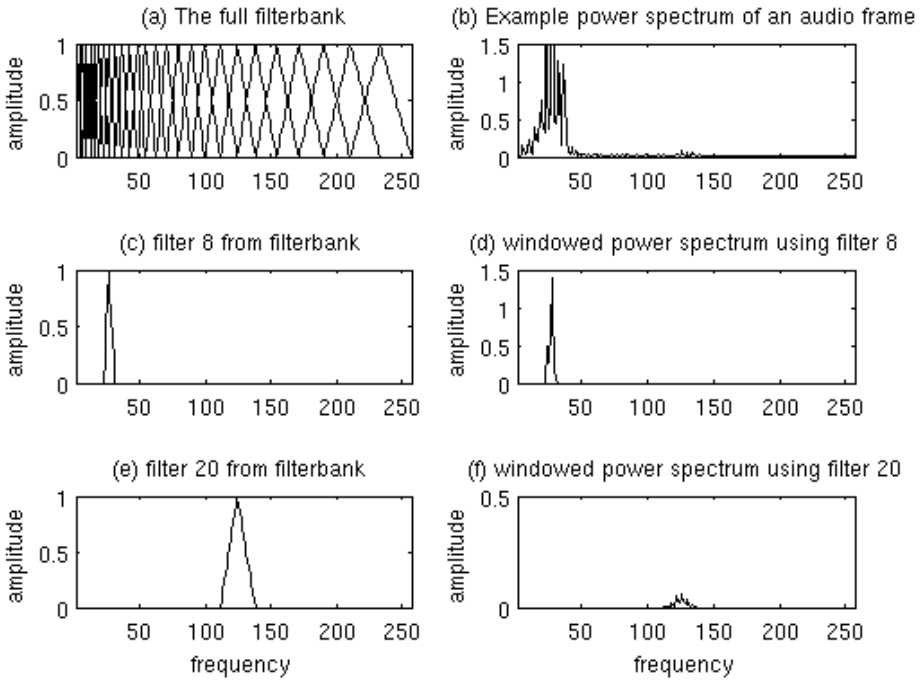


Fig 03, This is the Plot of Mel Filter-bank and windowed power spectrum. Here, one can see that analyzing the data and taking out features would be easier when we have power spectral which can tell us about the frequency exist in given speech/frames.

Then compute the Mel-spaced filter-bank. This is a set of 20-40 triangular filters that we apply to the periodogram power spectral. Our filter-bank comes in the form of 26 vectors of length 257. The set 26 triangular filter is standard for calculating male-spaced filter-bank. The calculation mechanism of filter-banks is provided as follows on the mfcc tutorial [5]. To take the Discrete Fourier Transform of the frame, perform the following:

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N} \qquad 1 \leq k \leq K \qquad \text{..... (3)}$$

where $h(n)$ is an N sample long analysis window (e.g. hamming window), and K is the length of the DFT. The periodogram-based power spectral estimate for the speech frame $s_i(n)$ is given by:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \qquad \text{..... (4)}$$

This is called the Periodogram estimate of the power spectrum. We take the absolute value of the complex Fourier transform and square the result. We would generally perform a 512-point FFT and keep only the first 257 coefficients as described in [5].

“in order to get filter-banks energies, multiply each filter-bank with the power spectrum, then add up the coefficients. Once this is performed we are left with 26 numbers that give us an indication of how much energy was in each filter-bank.

Now we create our filter-banks. The first filter-bank will start at the first point, reach its peak at the second point, then return to zero at the 3rd point. The second filter-bank will start at the 2nd point, reach its max at the 3rd, then be zero at the 4th etc. A formula for calculating these is as follows:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leq k \leq f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \leq k \leq f(m+1) \\ 0 & k > f(m+1) \end{cases} \qquad \text{..... (5)}$$

where M is the number of filters we want, and $f()$ is the list of M+2 Mel-spaced frequencies. All these above explanations are borrowed from [5].

Forth step would be to take the log of each of the 26 energies from previous step. This leaves us with 26 log filter-bank energies. Thereafter take the Discrete Cosine Transform (DCT) of the 26-log filter-bank energies to give 26 cepstral coefficients. For ASR (automatic speech recognition), only the lower 12-13 of the 26 coefficients are kept. The resulting features (12 numbers for each frame) are called Mel Frequency Cepstral Coefficients" [5]. The implementation of speech feature extraction can be find on this website mentioned in [6] but this implementation is in MATLAB Language but out implementation is in Python and can be seen on my GitHub Account [7].

7.2 Librosa for feature extraction

Librosa is a very well known in the field of MIR (Music information retrieval) now-a-days. It has proven it's significant in the area of speech feature extraction. The library documents are well documented and easy to understand. Here, is the YouTube tutorial/conference by author to learn more about the library [8]. And the library source code can be found on their GitHub account [9]. Also, one can see there on GitHub that the Librosa is up-to-date and they kept evaluating the changes requires. By using this library to extract features we were able to get total five features (mfccs, chroma, mel, contrast, tonnetz). Here is s small snippet of python script to understand it better.

```
def extract_feature(file_name):
    X, sample_rate = librosa.load(file_name)
    stft = np.abs(librosa.stft(X))
    mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axis=0)
    chroma = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T, axis=0)
    mel = np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T, axis=0)
    contrast = np.mean(librosa.feature.spectral_contrast(S=stft, sr=sample_rate).T, axis=0)
    tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(X),
                                              sr=sample_rate).T, axis=0)
    return mfccs, chroma, mel, contrast, tonnetz
```

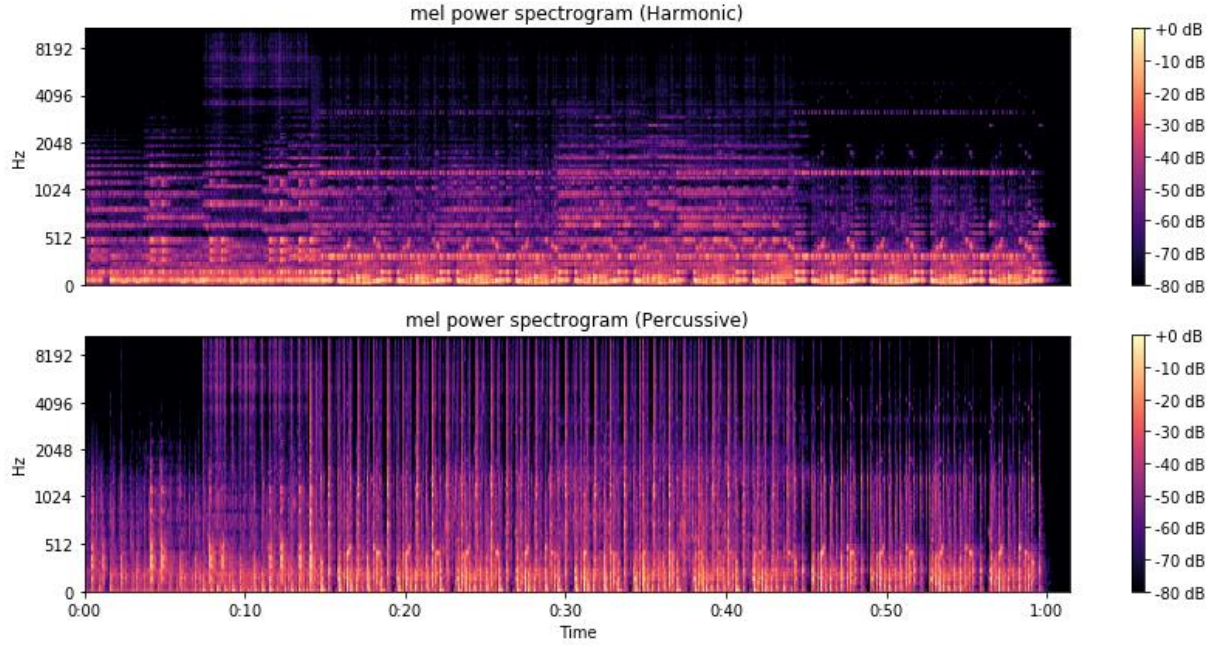
Here, you can see that the function names extract_feature will return mfcc, chroma, mel, contrast & tonnetz for a given speech. A speech or audio will be represented as a one-dimensional array or more specifically NumPy arrays, denoted as y here in the implementation. Typically, the signal y is accompanied by the sampling rate (denoted by sr = sample_rate) which denotes the frequency (in Hz) at which values of y are sampled. The duration of a signal can then be computed by dividing the number of samples by the sampling rate.

By default, when loading stereo audio files to the librosa, the librosa.load () function downmixes to mono by averaging left- and right-channels, and then resamples the

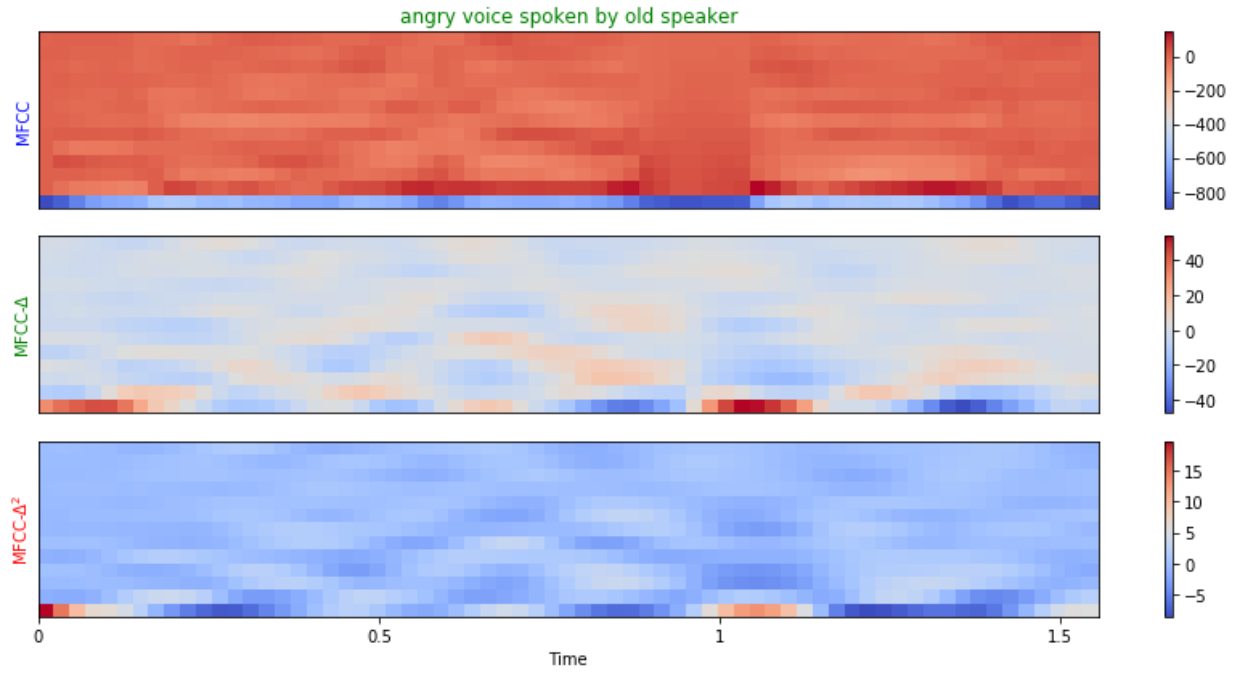
monophonic signal to the default rate sr=22050 Hz [10]. Since we are using default frame length therefore, the default frame and hop lengths are set to 2048 and 512 samples, respectively. At the default sampling rate of 22050 Hz, this corresponds to overlapping frames of approximately 93ms spaced by 23ms. If you want to change the sample rate you can choose `librosa.load(audio_path, sr=44100)` to resample at 44.1KHz, `librosa.load(audio_path, sr=None)` to disable resampling. As a output librosa produce two-dimensional results stored as NumPy.ndarray, e.g., $S[f, t]$ might contain the energy within a particular frequency band f at frame index t. We follow the convention that the final dimension provides the index over time, e.g., $S[:, 0]$, $S[:, 1]$ access features at the first and second frames. Feature arrays are organized column-major (Fortran style) in memory, so that common access patterns benefit from cache locality [10]. We can easily see the graphical representation of all these features and hoe different emotions have different statistics when it comes to feature extraction. Let's have a look for each and every feature that we have extracted.

7.3MFCC With Librosa

Here, is Mel-power spectrogram. 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 power cepstrum. This frequency warping can allow for better representation of sound - Wikipedia.

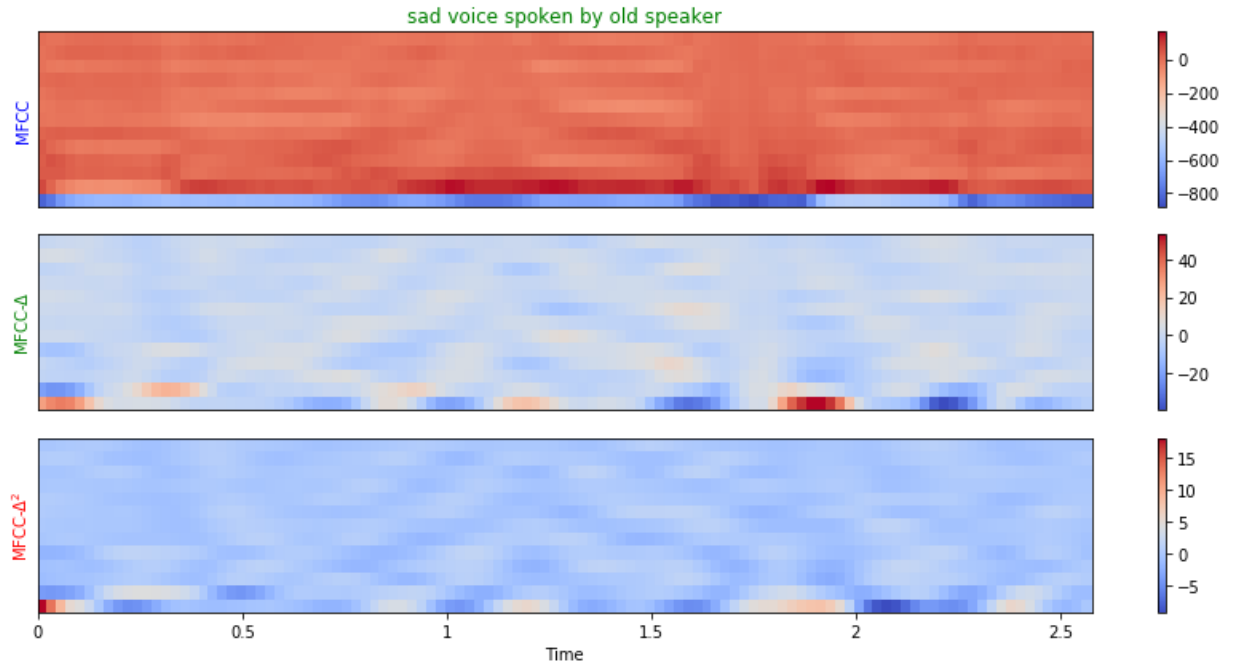


Mel-frequency cepstral coefficients are generally used to represent texture or timbre of sound. Below picture shows the mfcc pictorial data of angry voice, spoken by old speaker. The details are not much to be identified by normally looking. But still one can see that there are some details visible to get the idea and then compare it with different emotion holding speech (see below).



There can be variations on such process, for example: differences in the size, shape or spacing of the chosen windows used to map the scale or mixing up dynamics features like "delta" and "delta-delta". This delta meaning first order frame to frame difference coefficient and “delta-delta” is second-order frame-to-frame difference) coefficients.

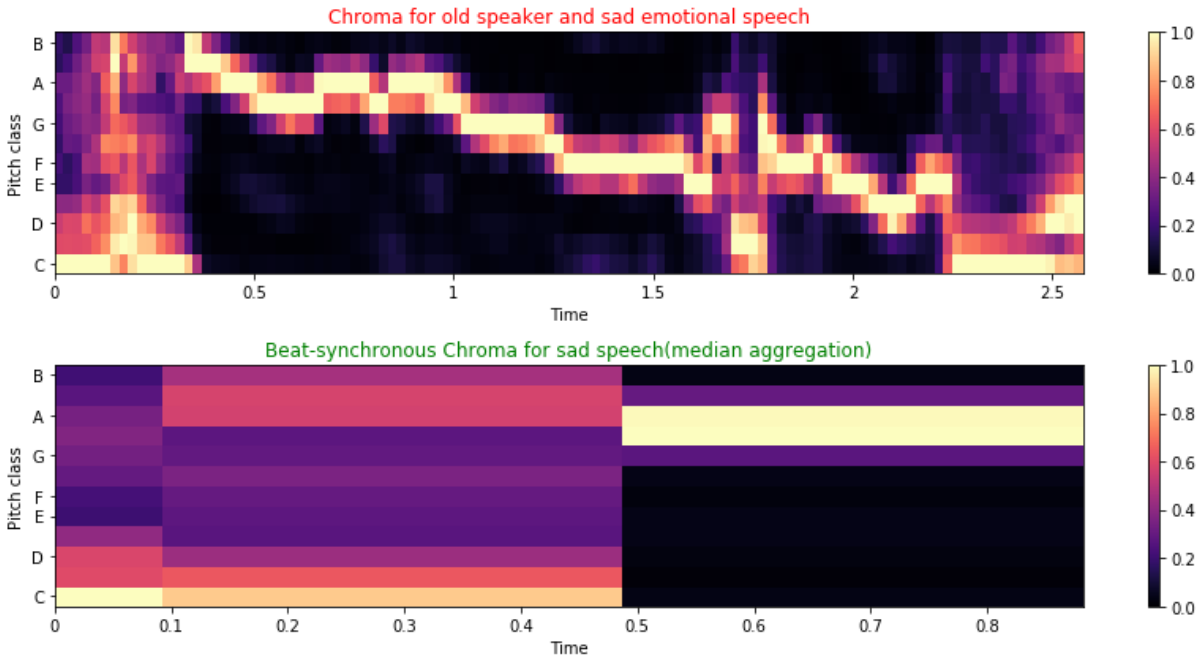
They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). In the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely. Such frequency warping can be used for better representation of sound - Wikipedia.



The above picture represents the mfcc details of sad voice spoken by old speaker. Again, it's pictorial representation so if there is little change/difference one can't identify but in numerical representation you can easily see the changes. Now let's see the other features extracted.

7.4 Chroma

Chroma known as Pitch class as well which representations are often used to encode harmony while suppressing variations in octave height, loudness, or timbre. Wikipedia has very good definition for it. "In the music context, the term chroma feature or chroma-gram closely relates to the twelve different pitch classes. Chroma-based features, which are also referred to pitch class profiles, are a powerful tool for analyzing music whose pitches can be meaningfully categorized (often into twelve categories) and whose tuning approximates to the equal-tempered scale. One main property of chroma features is that they capture harmonic and melodic characteristics of music, while being robust to changes in timbre and instrumentation." Librosa provides two kind of flexible pitch class implementations: one uses a fixed-window STFT analysis and the other uses variable-window constant-Q transform analysis (chroma_cqt) [10]. Instead of computing the mean delta-MFCC within each beat, let's do beat-synchronous chroma We can replace the mean with any statistical aggregation function, such as min, max, or median in chroma [9].



There is also one factor that we can not see the much variation in these pictures because the voice is really short and contain just one semantic utterance. Still in the above picture one can see that variation and the way it captures harmonic and melodic characteristics of speech. We saw how the display module provides simple interfaces to visualize audio data through python's matplotlib. The first function is to display the simple wave plot and then we have many other modules to display different statistics of the given speech or audio signal.

```
print('Estimated tempo for SRP speech sample:        %.2f BPM' % tempo)
print('First 5 beat frames:    ', beats[:5])
print('First 5 beat times:     ', librosa.frames_to_time(beats[:5], sr=sr))
```

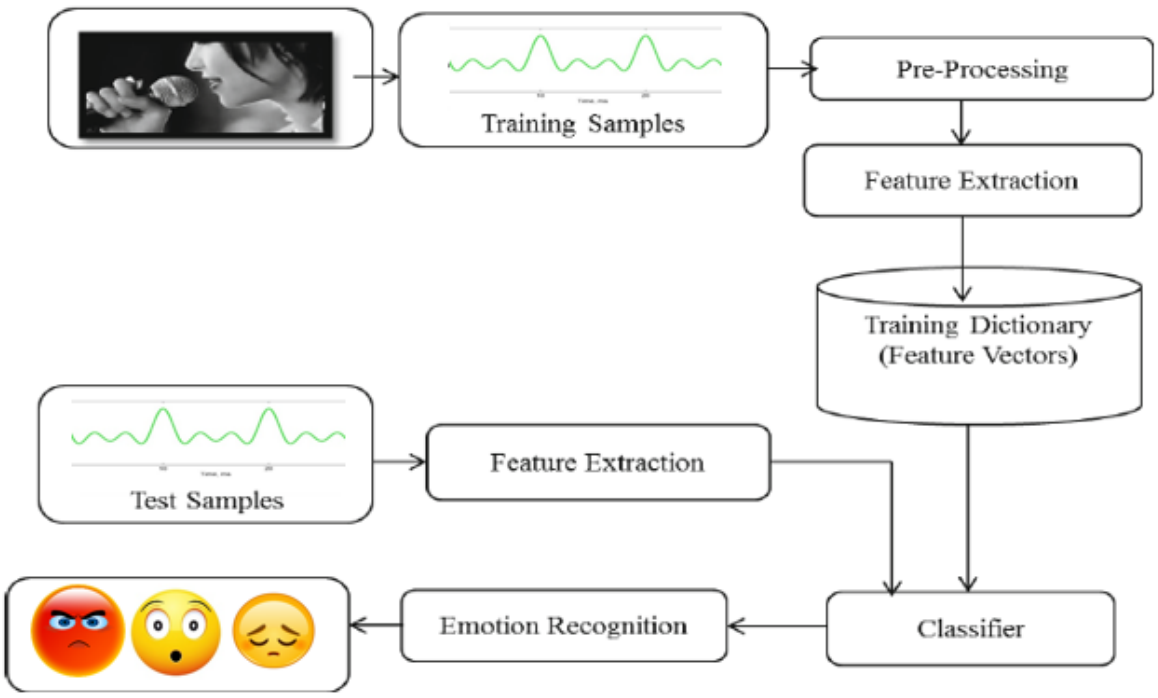
Here frame numbers are great and all, but when do those beats occur? We could also get frame numbers from times by librosa.time_to_frames(). Below are the outcomes of above script –

```
Estimated tempo for SRP speech sample:        143.55 BPM
First 5 beat frames:    [ 4 21 38]
First 5 beat times:     [ 0.09287982  0.48761905  0.88235828]
```

Remember all the statistics shown above is from single data-point. Meaning only one speech sample were used to get these pictures and statistics for understanding. We can of course print the numerical statistics of all the features shown above in pictorial form.

8 Model Architecture

Before we proceed to the classification and train and test accuracy we must have a look at the architecture of the whole model. Here, is a diagram showing each and every step involves in the process to detect emotion form given speech.



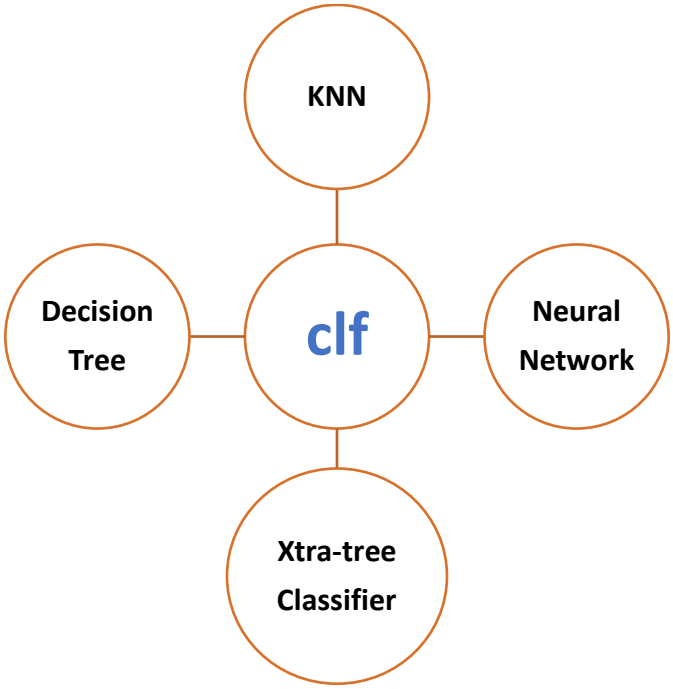
To illustrate this architecture, one can see that the input is simple voices/speeches. These voice samples belong to the training data-set then we might need to preprocess the speech that was not required in our case, since the data which was collected were completely pre-processed and clean. Next step would be to extract features from given samples and make a feature vector. Now we can use the extracted features to train the model with respect to feature & its emotion classes. Next, save the model weights for future use during testing. The next task would be to take test data-set and extract the features and use the classification model that we have saved before to get the detection of emotions. Something like this.

```
76  print("Input File: ", target_files[i], "|", " Predicted Emotion Is:", val)
```

```
Input File:  ./test_sounds_3\OAF_bite_neutral.wav |  Predicted Emotion Is: neutral
```

9 Classification Approaches

The feature extraction and data pre-processing are undoubtedly the crucial part of the project, but classification algorithm plays its significant as much as they do. This is the way to decide which speech samples belongs to which emotion class. In this section I am going to use some of popular machine learning classification algorithms and most of them are borrowed from Machine learning lecture provided by ISMLL, University of Hildesheim. Below are some classification mechanisms used for this research –



These are the beloved supervised machine learning algorithm that we will be using for the classification purposes. Apart from this, the model were tried with other classifiers aw well which did not give us good results (i.e. having poor classification accuracy) but one can see them in conclusion/finding section. Choosing the classifier also plays very important role to provide better prediction power. Here, we are dealing with Multi-Category Targets/Multi-class/Polychotomous Classification. Meaning prediction of a nominal target variable with more than 2 levels/values. And we have 7 classes in our case. The results and the model explanation in given below in detail.

9.1 K- Nearest Neighbor Classifier

Motivation behind using KNN was its simplicity and it is very basic model that will help us to find out what is the performance while working with initial model then we can go for the advanced and more latest algorithms such as Neural Networks. Moreover, the KNN algorithm is a powerful and flexible classifier that is frequently utilized as a benchmark for many mind impressive classifiers, for example, Artificial Neural Networks (ANN) and Support Vector Machines (SVM). In spite of its effortlessness, KNN can outperform capable classifiers and it has proved its significance by being utilized at many crucial projects. Such as, economic/ financial forecasting, data compression and genetics as well. KNN can be used for both the purposes regression and classification. Our objective is classification. Here are the steps involved in KNN classification algorithm.

1: **procedure** PREDICT-KNN-CLASS($q \in \mathbb{R}^M, \mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\} \in \mathbb{R}^M \times \mathcal{Y}, K \in \mathbb{N}, d$)
2: allocate array D of size N
3: **for** $n := 1, \ldots, N$ **do**
4: $D_n := d(q, x_n)$
5: $C := \text{ARGMIN-K}(D, K)$
6: allocate array $\hat{p}$ of size $\mathcal{Y}$
7: **for** $k := 1, \ldots, K$ **do**
8: $\hat{p}_{C_k} := \hat{p}_{C_k} + 1$
9: **for** $y \in \mathcal{Y}$ **do**
10: $\hat{p}_y := \frac{1}{K}\hat{p}_y$
11: **return** $(\hat{p})_{y \in \mathcal{Y}}$

To be more explanatory our task is to determine the *K* which is number of nearest neighbor and then calculating the distance between the query and all the training samples then sort the distance and determine nearest neighbor on the basis of *k-th* minimum distance. Next step would be to collect the category *Y* (target) and use majority of category of NN as the prediction value of the query instance. Our output would be emotion class membership (happy, sad, angry ....) where an object is classified by a majority vote of its neighbors, with the object being assigned to the emotion class most common among its *k* nearest neighbors.

There are some parameters that one need to optimize in order to get optimal results. Here some of default parameter which scikit-learn provides that can be modified easily to optimize the model to fir the data best.

```
clf= KNeighborsClassifier(n_neighbors=7, weights='uniform', algorithm='auto',
                          leaf_size=30, p=2, metric='minkowski',
                          metric_params=None, n_jobs=1)
```

Here, number of neighbors are set to 5 by default but for the sake of our project we set them to 7. Minkowski was used as distance metric. The default metric is minkowski, and with p=2 (equation below) is equivalent to the standard Euclidean metric.

$$\hat{p}(Y = y \mid x) := \frac{1}{K} \sum_{(x',y') \in C_K(x)} I(y = y') \qquad \text{..... (6)}$$

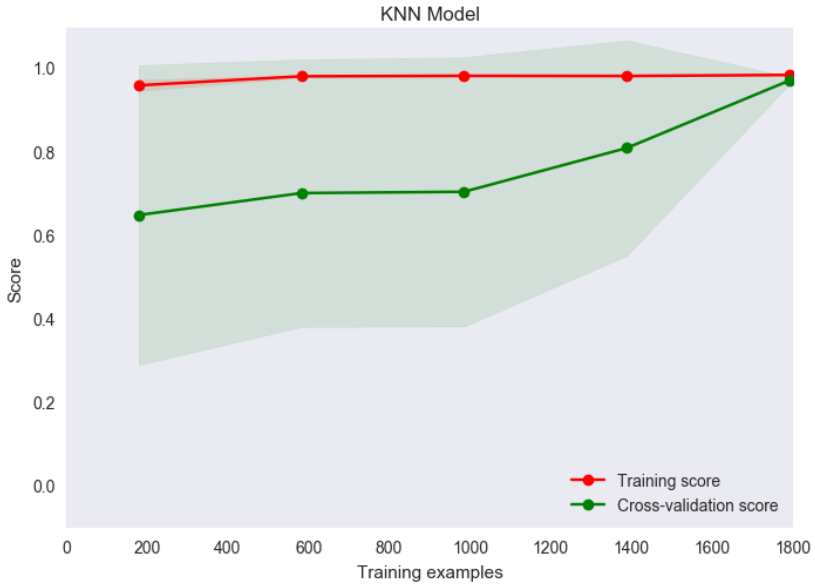
and then predict the class with maximal predicted probability

$$\hat{y}(x) := \arg\max_{y \in \mathcal{Y}} \hat{p}(Y = y \mid x) \qquad \text{..... (7)}$$

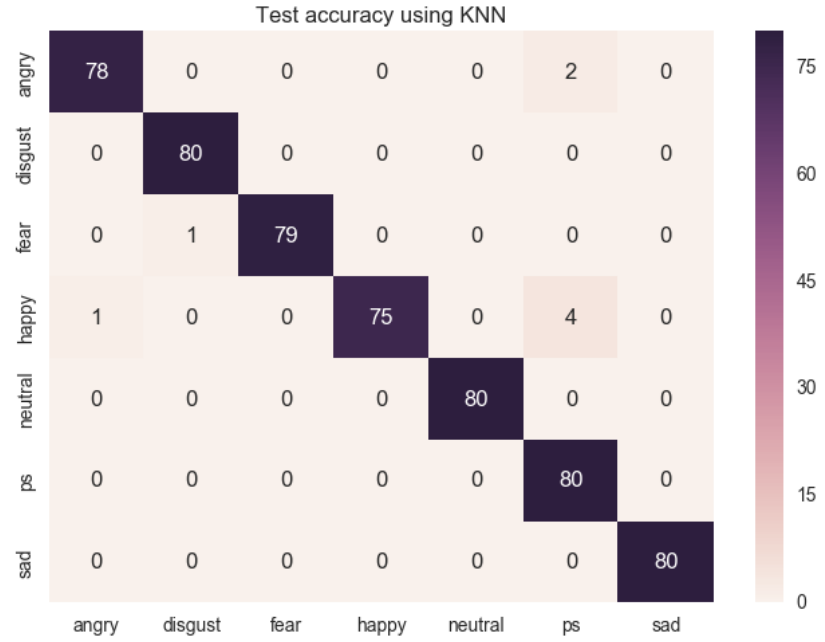
$$d(x, y) := \left(\sum_{i=1}^{n} (x_i - y_i)^2\right)^{\frac{1}{2}} \qquad \text{..... (8)}$$

(1) represents k-NN classification algorithms equation (2) shows how to predict class with maximal predicted probability (3) Shows Euclidean distance metric expression. Here is the full documentation (https://goo.gl/zZHfJu) to learn more about K-NN classification parameters.

Let's look at the results during training phase and then we will go to testing phase.



The training accuracy was 0.9857 and one can see the *graph 01* to observe learning curve. Here the training sample are at the x-axis and accuracy score is at y-axis. Here the samples are shown till 1800 but we have 2240 speeches for training phase.



Confusion Matrix 01. This is the accuracy score during testing on 560 speech samples. On the x-axis we have predicted labels and, on the y-axis, we have True labels. Therefore, one can identify that 4 happy and 2 angry speeches classified as pleasant surprise and 1 happy speech was misclassified as angry. Apart from this 1 fear speech got miss-classified as disgust.

```
Accuracy Score: 0.985714285714
Number of correct prediction using KNN: 552 out of 560
```

The training accuracy was pretty good and model perform very well too during testing phase. It got the accuracy score of 98%. It has 8 miss-classification. Note - If it is found that two neighbors, neighbor '*k+1*' and '*K*', have identical distances but different labels, the results will depend on the ordering of the training data.

9.2 Decision Tree Classifier

Another well known algorithm is Decision Tree. Decision Trees (DTs) are a non-parametric supervised Machine learning method used for classification and regression too. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the speech features (mfcc, chroma, tonnetz & etc.) that we have extracted before. A decision tree is a tree that at each inner node has a splitting rule that assigns instances uniquely to child nodes of the actual node, and at each leaf node has a prediction *Y* (emotion class). Each branch of a decision tree can be formulated as a single conjunctive rule. Such as,

if condition$_1$(x) and condition$_2$(x) and . . . and condition $_k$(x),

then y = class label at the leaf of the branch.

In our case the condition is nothing but the features. Meaning, if the mfcc, chroma, tonnetz etc... is *x* then the given speech belongs to *y* class. Here every feature is a condition that will be considered during decision making (assigning speech *x* to a class *y*). Our split is multivariate since we have many features to play role into decision making.

```
1: procedure PREDICT-DECISION-TREE(node T, instance x ∈ ℝ^M)
2:     if T.split ≠ ∅ then
3:         z := T.split(x)
4:         T' := T.child[z]
5:         return PREDICT-DECISION-TREE(T', x)
6:     return T.class
```

Here is the pseudocode to learn decision tree. In the pseudocode at line 4:- T' is nothing but to keep an array that maps all the possible outcomes of the split to the corresponding child node. Below are the parameters and hyperparameters that were used for this project. There are many other benefits to use decision tree such as it's Simple to understand and to interpret. The tree architecture can be visualized too by using *graphviz.* Decision Tree uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by Boolean logic. By contrast, in a black box model like deep learning's neural network maybe more complex to interpret. It can be unstable too because small variations or changes in the data might result in a completely different tree/decision being generated.

Here, are the parameters and hyperparameters that were used for this project - Criterion: (default="gini") The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gini entropy is a way to measure the impurity expressed as –

$$Entropy = -\sum_j p_j \log_2 p_j$$

..…… (9)

Gini index (a criterion to minimize the probability of misclassification):

$$Gini = 1 - \sum_j p_j^2$$

...…… (10)

Splitter: The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split we have preferred ‘best’ splitter.

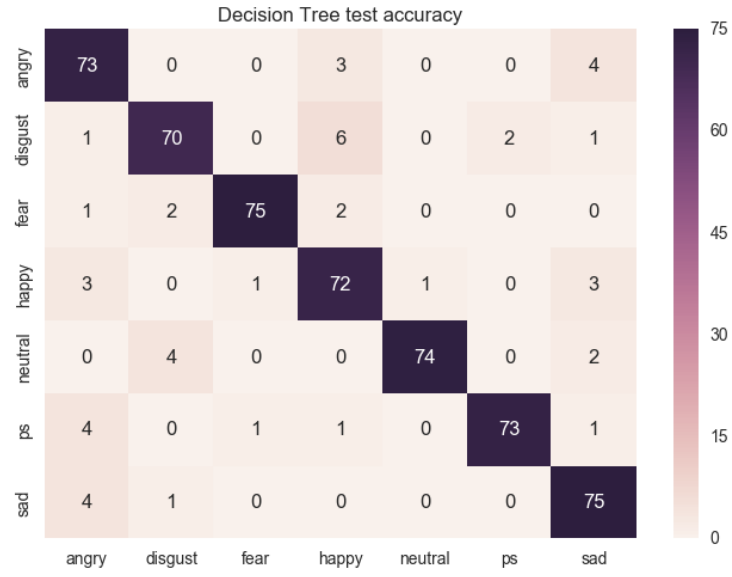
Max_depth: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than *min_samples_split* samples. For the implementation we have used mostly default parameters provided by scikit learn python library.

```
clf = tree.DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None,
                                  min_samples_split=2, min_samples_leaf=1,
                                  min_weight_fraction_leaf=0.0, max_features=None,
                                  random_state=None, max_leaf_nodes=None,
                                  min_impurity_decrease=0.0, min_impurity_split=None,
                                  class_weight=None, presort=False)
```

Above you can see the snippet of the decision tree classifier used for the project. The parameters and hyperparameters could be tuned more to get better results. The above parameters are the default provided by scikit-learn.

Now we can see the results by using decision tree. The number of training and test set are same and will remain same through-out the research.

Learning curve 02. Here is the outcome of decision tree. The first picture shows the learning curve during training and one can see the training and validation error.



Here, is the *confusion matrix 02* which shows accuracy during test. Again, on the y axis we have true labels and on the x axis we have predicted one. One can see that there are many miss-classification.
Such as, 10 disgust speeches are miss-classified and many others too. Altogether, we have 48 miss-classification. And the accuracy score is 91%.

```
Accuracy Score of Decision Tree model is: 0.914285714286
Number of correct prediction by using Decision Tree is: 512 out of 560
```

The score of prediction can be optimize by setting the optimal parameters and hyperparameters.

9.3 Neural Network Classifier

There are many definitions for neural network but here is the one which I like personally.

“An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.” [11]

The neural network model has proven its significance in leaning from data and having better prediction capabilities. Other advantages include, Adaptive learning & Fault Tolerance etc.

Network Architecture -

Here is the neural network architecture which was used to achieve our objective.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 100)               19400
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 20)                1020
_________________________________________________________________
dropout_3 (Dropout)          (None, 20)                0
_________________________________________________________________
dense_4 (Dense)              (None, 7)                 147
=================================================================
Total params: 25,617
Trainable params: 25,617
Non-trainable params: 0
_________________________________________________________________
```

The above picture shows the model summary of the network. Meaning you can see number of layers, types of layers & drop-out layer were used through-out the network along with some statistics about parameters too.

There are several ways to implement neural networks. Such as, TensorFlow, Theano, CNTK & Keras. I have chosen Keras over other platforms. There are many advantages of using Keras. Such as, it allows for easy and fast prototyping (through user friendliness, modularity, and extensibility). Supports both kind of popular layers. Such as, convolutional networks and recurrent networks, as well as mixture of both. It runs seamlessly on normal CPUs and powerful GPUs too. The above architecture drawn from below script.

```
def baseline_model():
    deep_model = Sequential()
    deep_model.add(Dense(100, input_dim=193, activation="relu", kernel_initializer="uniform"))
    deep_model.add(Dropout(0.5))
    deep_model.add(Dense(50, activation="relu", kernel_initializer="uniform"))
    deep_model.add(Dropout(0.5))
    deep_model.add(Dense(20, activation="relu", kernel_initializer="uniform"))
    deep_model.add(Dropout(0.5))
    deep_model.add(Dense(7, activation="softmax", kernel_initializer="uniform"))

    deep_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return deep_model
```

Parameters, Hyperparameters and Regularization -

Epochs = 100
Batch Size = 25
Optimizer = Adam
Loss = We calculate a separate loss for each class label per observation and sum the result.

$$-\sum_{c=1}^{m} y_{o,c}\log(p_{o,c})$$

M - number of classes (angry, happy, sad…)
y - binary indicator (0 or 1) if class label c is the correct classification for observation o
p - predicted probability observation o is of class c

In order to regularize the model, we limit the capacity of a model to avoid over-fitting. We extend the cost-function by adding a penalization term –

$$\tilde{J}(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(\theta) \quad \ldots\ldots\ldots\ldots (11)$$

Here alpha also known as the regularization penalty. We regularize the neuron weights, but not the bias terms. Here, drop-out is to regularize the model.

Dropout a node by multiplying its output by zero only input and hidden nodes are dropped out. For input/hidden units typically, an input unit is included with a probability of 0.8 and hidden unit with a probability of 0.5. Drop-out for forward computation read as –

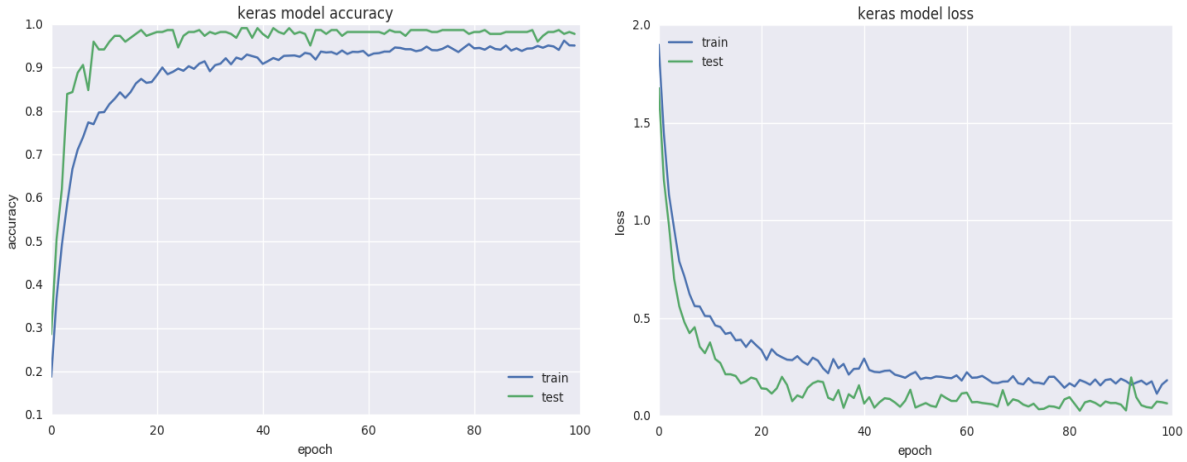
$$
\begin{aligned}
h^{(1)} &= g^{(1)}\left(W^{(1)^T}\left(x \otimes \mu^{(0)}\right) + b^{(1)}\right) \\
h^{(2)} &= g^{(2)}\left(W^{(2)^T}\left(h^{(1)} \otimes \mu^{(1)}\right) + b^{(2)}\right) \\
&\cdots \\
h^{(L)} &= g^{(L)}\left(W^{(L)^T}\left(h^{(L-1)} \otimes \mu^{(L-1)}\right) + b^{(L)}\right)
\end{aligned}
\quad \ldots\ldots\ldots\ldots (12)
$$

Here $\otimes$ is the element-wise multiplication. μ is drop-out mask for input and hidden layer which is –

$$\mu^{(0)} \sim \text{Bernoulli}(p_{\text{input}})_N$$

$$\mu^{(l)} \sim \text{Bernoulli}(p_{\text{hidden}})_{N_l}$$

Let's look at the result obtained by neural network in emotion detection from speech.



The *learning curve 03* above shows the 1) accuracy vs epochs 2) loss vs epochs. Remember this graph shows learning curve during training phase. By the graph we can see that the model learned really quickly in roughly 40 epochs.

One can see the loss and accuracy in exact numerical form too.

```
Epoch 98/100
1259/1259 [==============================] - 0s 202us/step - loss: 0.0981 - acc: 0.9603 - val_loss: 9.9896e-05 - val_acc: 1.0
000
Epoch 99/100
1259/1259 [==============================] - 0s 214us/step - loss: 0.1081 - acc: 0.9508 - val_loss: 1.9593e-04 - val_acc: 1.0
000
Epoch 100/100
1259/1259 [==============================] - 0s 213us/step - loss: 0.1448 - acc: 0.9476 - val_loss: 0.0010 - val_acc: 1.0000
```

Above is the statistics about the last three epochs where one can see the loss validation accuracy etc.… The time to train this network was not much but approximately 20 minutes on dual core CPU with 8 GB of ram. Now we can see the test accuracy and how powerful the prediction is –

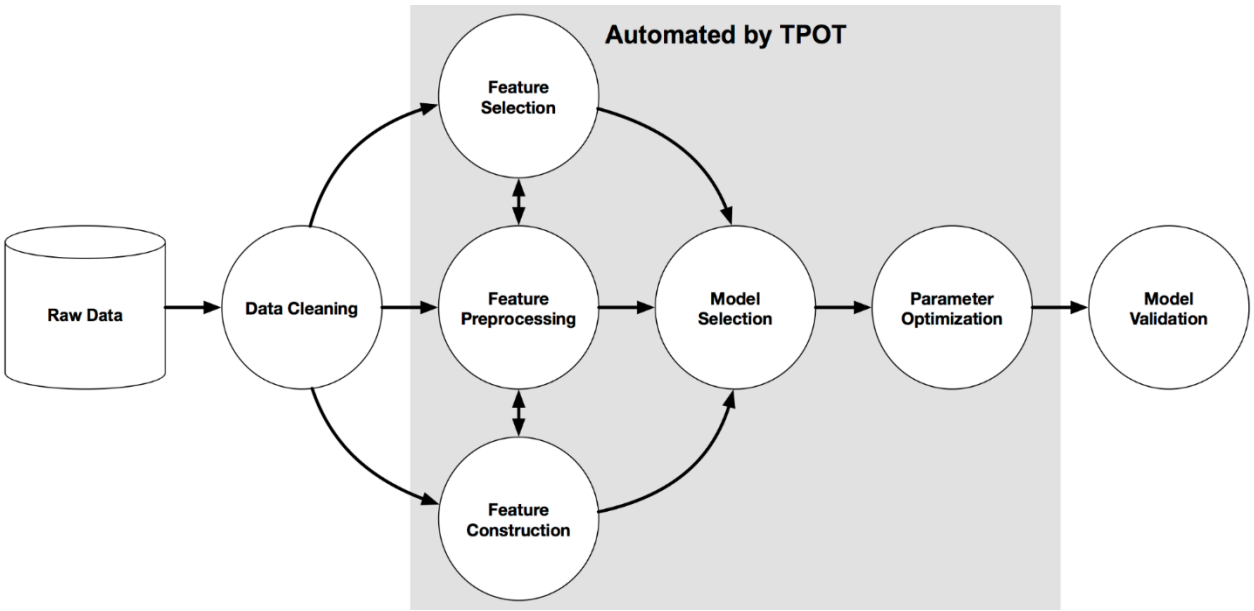
Confusion Matrix 03. Here, we have witnessed 99% of prediction correctly. Only 2 speeches got miss-classified those were angry and disgust samples and landed into pleasant surprise and sadness respectively. Again, for the simplicity y-axis has true label and x-axis has predicted one.

```
Accuracy Score: 0.996428571429
Number of correct prediction: 558 out of 560
```

The result shows that the neural network has really good potential to predict correct emotion classes. The obtained result missed only 2 speeches else 558 speeches were classified in right class.

10 AutoML (Autonomous Machine Learning)

The project has used the privilege to keep AutoML into consideration. Consider TPOT your Data Science Assistant. TPOT is a Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming. AutoML is nothing else but a platform which can suggest you best classifier along with the optimal parameters to best fit the data. Here is some very good explanation given by the official web-site of AutoML TPOT [12]. Also, there is other platform too to get such information to choose best classifier and parameter but because of tpots simplicity we go with it. TPOT will automate the most difficult and time-consuming part of machine learning by intelligently exploring thousands of possible pipelines to find the best one for our emotion speeches. Here is the complete architecture of automated machine learning pipeline tpot.

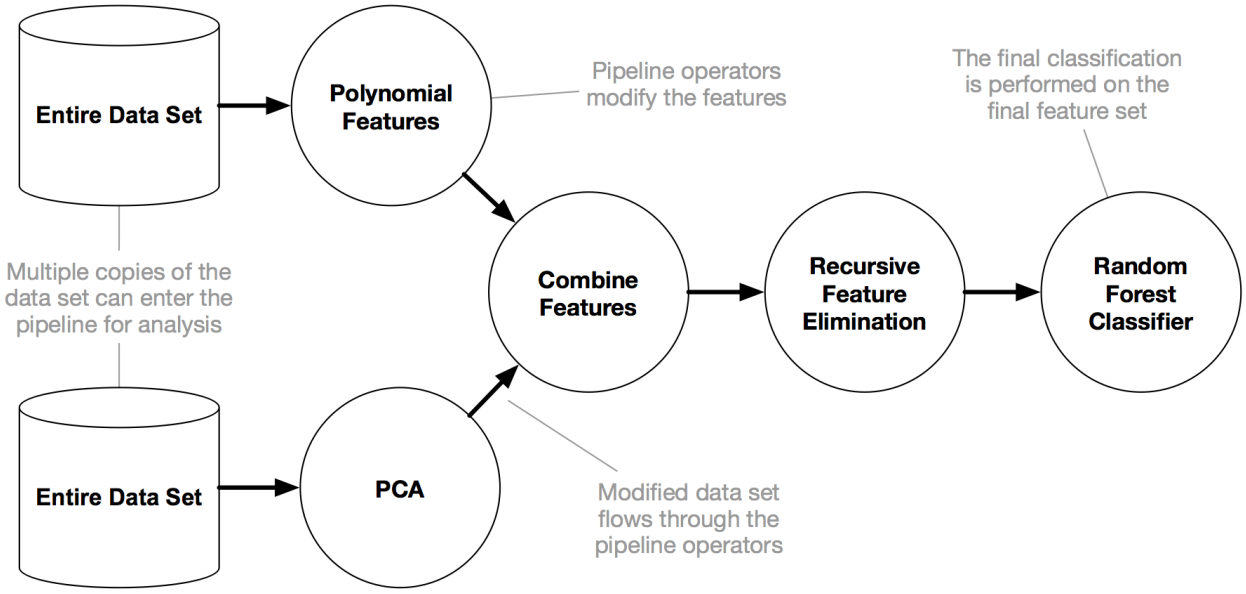


The above picture is representing machine learning pipeline. The part shows in dark color will be done automatically by AutoML tpot. Which includes feature selection, feature pre-processing, feature construction, model selection & parameter optimization. Tpot takes too much time to give you the outcome since it looks for several model combinations (random forests, linear models, SVMs, etc.) to provide us the best fit for our data-set. Think about the Grid-Search which takes hours or maybe days to run. Any AutoML algorithms is not intended to run for few minutes. Of course, you can run TPOT for only a few minutes and it will find a reasonably good pipeline for your dataset. However, if you don't run TPOT for long enough, it

may not find the best possible pipeline for your dataset. It may even not find any suitable pipeline at all, in which case following error will be raised.

```
RuntimeError('A pipeline has not yet been optimized. Please call fit() first.')
```

Often it is worthwhile to run multiple instances of TPOT in parallel for a long time (hours to days) to allow TPOT to thoroughly search the pipeline space for your dataset. Once TPOT is finished searching (or you get tired of waiting), it provides you with the Python code for the best pipeline it found so you can tinker with the pipeline from there [12].



It's important to realize why Tpot takes this much time to run. With the default TPOT settings (100 generations with 100 population size), TPOT will evaluate 10,000 pipeline configurations before finishing. Now if you will do the same thing with grid search having 10,000 model settings or configuration and if you are using 10-fold cross validation, roughly it will be 100,000. Typical TPOT runs will take hours to days to finish (unless it's a small dataset), but you can always interrupt the currently running program and partway through and see the best results optimize by Tpot till that moment. TPOT also provides a `warm_start` parameter that lets you restart a TPOT run from where it left off [12]. Most of the above explanation are borrowed from Tpot official web-site.

10.1 ExtraTreesClassifier

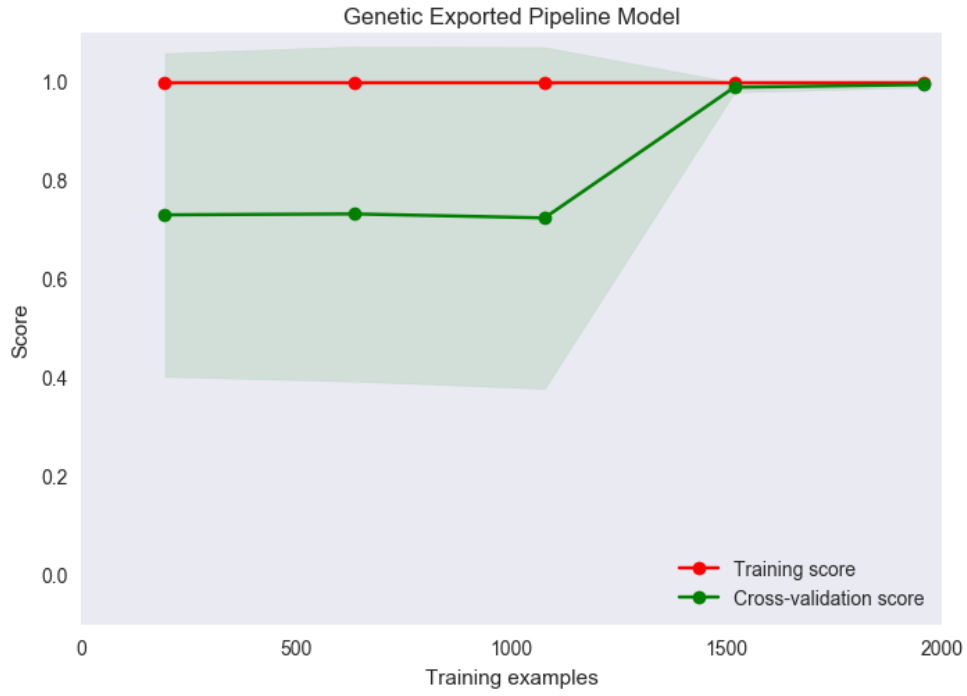
What we have got when we used Tpot for the suggestion of best machine learning pipeline? In our case for this emotion detection dataset it recommended us ExtraTreesClassifier. There are some confusions between extra tree classifier and random forest. Here is the thin line of difference. Extra-Tree method produces piece-wise multilinear approximations, rather than the piece-wise constant ones of random forests. Here is what we chose as a parameter for extra tree classifier –

```
exported_pipeline = TPOTClassifier(generations=5, population_size=20, cv=5,
                                   random_state=42, verbosity=2)
```

Here is what we chose as a parameter for extra tree classifier –

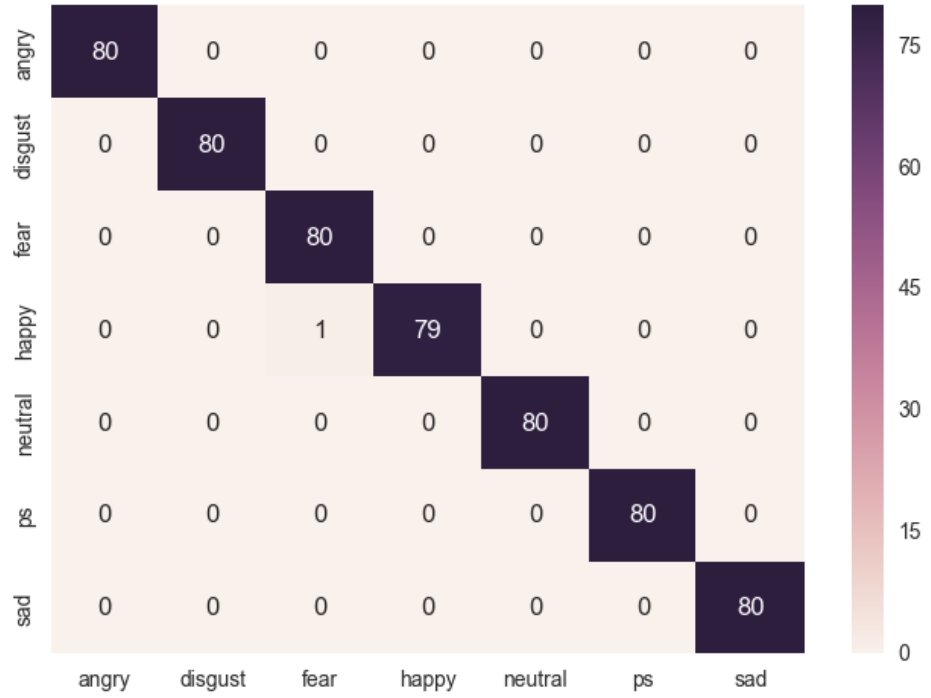
```
exported_pipeline = ExtraTreesClassifier(bootstrap=True,
                                         criterion="entropy",
                                         max_features=0.25,
                                         min_samples_leaf=1,
                                         min_samples_split=8,
                                         n_estimators=100)
```

Let's go straight to the results obtained by Tpot suggested model combination during training phase first.



The *learning curve 04* shows performance of the model in learning during training phase. On the x-axis we have training samples and on the y-axis we have score.

Now we will see the performance during testing phase. In the *confusion matrix 04* below we can see that the AutoML worked really impressive in creating best pipeline and model configuration. ExtraTreesClassifier achieved 0.9982 accuracy in predicting correct emotion class. This model overtakes all the other models and win the race.




```
Accuracy Score: 0.998214285714
Number of correct prediction: 559 out of 560
```

This model missed just 1 speech that was happiness and it classified it as Fear, which is weird since both are opposite emotions. I mean if the speech is happy and it got classified as pleasant surprise then it would be understandable since these both emotions are a bit similar.

11 Conclusion

Table below consist all the model and their performance on Toronto speech set in order to predicting the emotion from given speech samples. Here one can see that AutoML model outperform each and every model by achieving 99.82% of accuracy score. Decision tree has the worse result among all 4 models. The reason could be the parameters were not optimized at all. All the default parameter and hyperparameters provided by pythons scikit learn implementation were used as it is.

No.	Model Name	Performance (%) Train/Test
01	KNN	98.57/98.57
02	Decision Tree	1.0/91.42
03	Neural Network	97.77/99.64
04	ExtraTreesClassifier	99.64/99.82

Table 03

During experiment SVM with linear and rbf kernel were tried by ensembling it but didn't perform well at all. I have noticed the accuracy approximately 50% when used ensemble of svm with both rbf and linear kernel. Moreover, MLP has poor predicting power in this research too.

Since we have speech data collected by two speaker whose age were 24 and 64 respectively. Therefore, I had a chance to see how much young emotion speech translate or old person speech or vice versa but obtained disappointing results. As far as I remember the accuracy was around 45%. It means that old and young individual speech varies when it comes to emotion. Or other reason could be the data-set. There are many research going on where people are discussing and debating that how much acted emotion translate to real emotion and the data-set we have used is followed acted way to collect it. The future work of such research could be real time emotion detection and deployment of such system into production.

12 Acknowledgment

I would like to appreciate consistent support from Prof. Schmidt-Thieme. I am so very grateful that he encouraged me to pursue such an exciting and hot research topic for my Student Research Project. The guidance fom my research mentor Mrs. Lydia Voss and whole ISMLL team can’t be forgettable. Mr. Mohsan Jameel also plays a great role as a guide in preparing poster and slides for conference. Heartily thanks to him. Apart from this, especial acknowledgement to my friend Manish Mishra and others who gave me great support and advise.

Thanks a Ton

13 References

[1] Emotion in speech: recognition by younger and older adults and effects on intelligibility (http://hdl.handle.net/1807/31738)

[2] Techniques for feature extraction in speech recognition system: a comparative study (https://goo.gl/1WyHBg)

[3] Emotion in speech: recognition and application to call centers (Valery A. Petrushin)

[4] Emotion Speech Corpus (https://tspace.library.utoronto.ca/handle/1807/24501)

[5] MFCC Tutorial (https://goo.gl/kuYLd8)

[6] Mfcc implementation in MATLAB (https://labrosa.ee.columbia.edu/matlab/rastamat/)

[7] Python Implementation of this project along with complete project

(https://github.com/nirajdevpandey/Emotion-Detection-from-Speech)

[8] Librosa library tutorial by author (https://www.youtube.com/watch?v=MhOdbtPhbLU)

[9] Librosa Library Source code (https://github.com/librosa/librosa)

[10] librosa: Audio and Music Signal Analysis in Python [Brian McFee at. el. 2015]

[11] Christos Stergiou and Dimitrios Siganos neural network tutorial (https://goo.gl/W3B3fz)

[12] AutoML TPOT official documentation (http://epistasislab.github.io/tpot/)

[13] AutoML TPOT official source code (https://github.com/EpistasisLab/tpot)

[14] L. Deng and D. Yu, "Deep learning for signal and information processing," in FTML, 2013.

[15] Real vs. acted emotional speech - Janneke Wilting, Emiel Krahmer & Marc Swerts (https://goo.gl/owVrtB)