SMAI

Speaker recognition - Comparative study

Final Project Report

Vishnu M. - 201556111
Navya Khare - 201564220
Tanmay Joshi - 201556069
Shubhangi Gautam - 201564189

Introduction

Each person has a distinct way of speaking. There are minute differences between people while they speak, this is used by humans to determine who is speaking. The task of Speaker Identification/Recognition involves the Identification of the speaker of an uttered sentence based on the characteristics of their voice.

Goal

The goal of this project was to implement Speaker Identification using various types of classifiers and make a comparative study on the results of each.

Dataset

For this project, the primary dataset that we have used was the TIMIT dataset, which consists of American English speakers of both genders, and 8 Dialect Region groups. This was commissioned by DARPA, recorded at Texas Instruments (TI) and transcribed at MIT, hence the name TIMIT[1].

Changes made to the dataset

For the purposes of our project, we had consolidated all the speakers of the various Dialect Regions into one. Since our task was speaker Identification and not Dialect Identification.

We have also made variable size subsets of the dataset with 5, 10, 15, 20, 25, and 50 speakers each.

Reasoning to choose TIMIT

We had chosen TIMIT dataset primarily because of its easy of use and the quality of the dataset, moreover this was a tried and tested dataset. There were other available datasets which relied on Youtube audio clips, the trouble with using these was data sanitisation, which would have taken much longer.

Overview of the dataset

Total Speakers -

- Varied between 5, 10, 15, 20, 25 and 50.

Number of sentences Uttered - split between Train and Test - 7:3

- Common Sentences = 2
- Independent Sentences = 8

All tracks were in NIST Sphere format and were around 5 seconds long.

Data Preprocessing

- Every audio file was converted from the original NIST SPHERE format to the normal .wav format.
- Then all the folders from each Dialect region were consolidated into a single folder.
- Data was split between Train and Test folders. Thereafter, the feature extraction pipeline was executed.

Feature Extraction

After getting the data into the format that we required, we needed to generate a feature vector for each of the sound file. This was accomplished as follows

Voice Activity Detector (VAD)

The purpose of this step was two-fold -

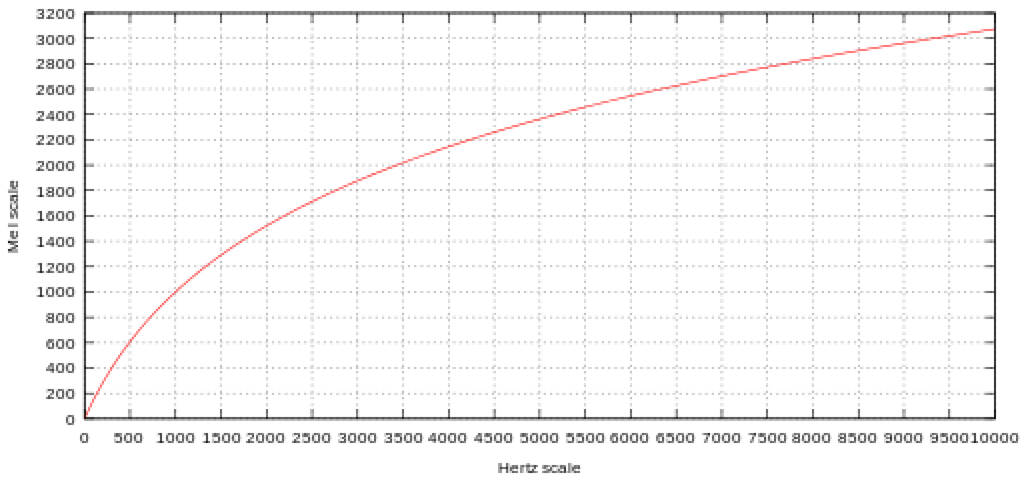
- Noise suppression - The removal of all the noisy parts of the signal
- Voice detection - The removal of all the silent parts of the signal where there is no voice uttered.

- ○ Frame size = 30ms
- ○ Threshold for silence = 0.01 times the average energy of the sound signal

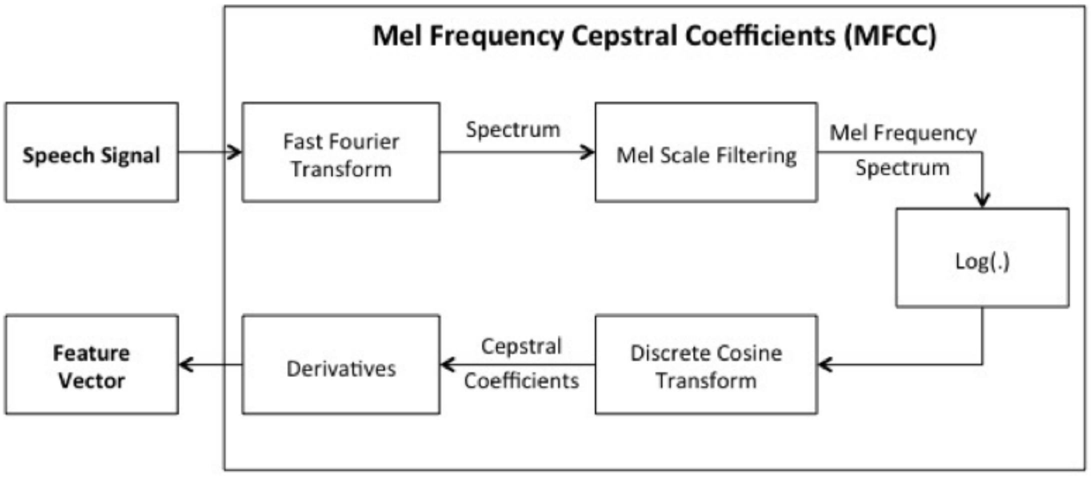
Both the above are important so as to train the classifiers to identify the speakers and not the environment. This process removes all the ambient sounds from the signal and also reduces the size of the signal by removing all the silent parts of the signal.

Mel Frequency Cepstral Coefficients -

In the Mel Scale, pitches are organised in such a way as to have equal distance between 2 pitches that are perceived by humans to be nearly identical.



*"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."*[2]



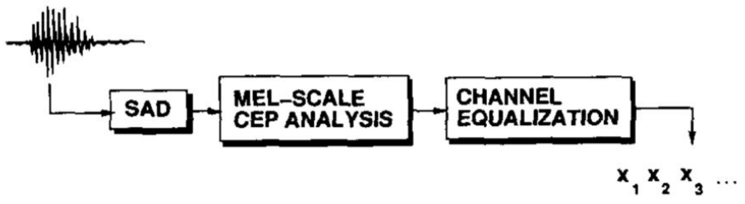
All the Coefficients that make up this MFC are known as the Mel Frequency Cepstral Coefficients (MFCC). MFCC algorithms are very receptive to noise in the input signals, therefore performing a VAD is necessary before creating a Feature Vector from it (However TIMIT dataset is pretty much noise free, so it might not have as much of an effect).

Implementation -

We have used MFCC extraction through sklearn talkbox module, which returns a size *n*13* matrix of coefficients, where n is a variable quantity that varies with the sound file.

The 13 fields are related to the power spectrum coefficients for each frame of the audio. We then had to normalize all these variable length matrices to a uniform size per audio file, which was accomplished by taking the mean of every one of the 13 coefficients across every frame of the audio file.

Result - Every audio file yields a **13 Dimensional Feature vector plus a Speaker label.**



Classifiers and Results-

We have implemented 4 different classifier with varying parameters, namely GMM, KNN, SVM and Neural Network.

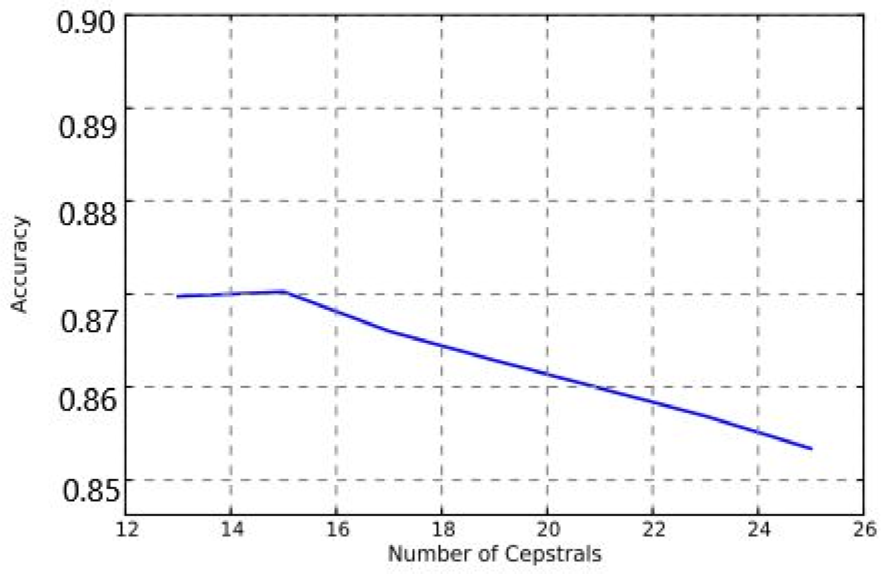
The raw results of each of the classifier can be found at this link.

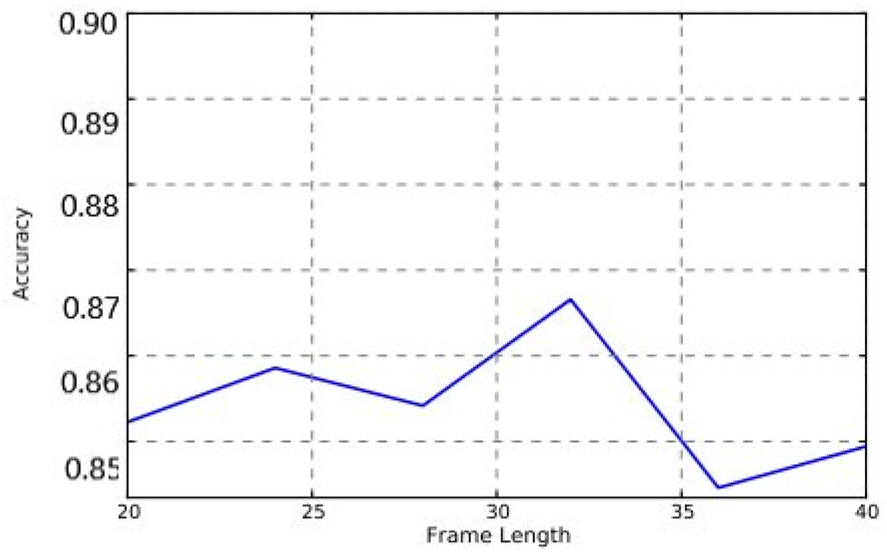
Preprocessing

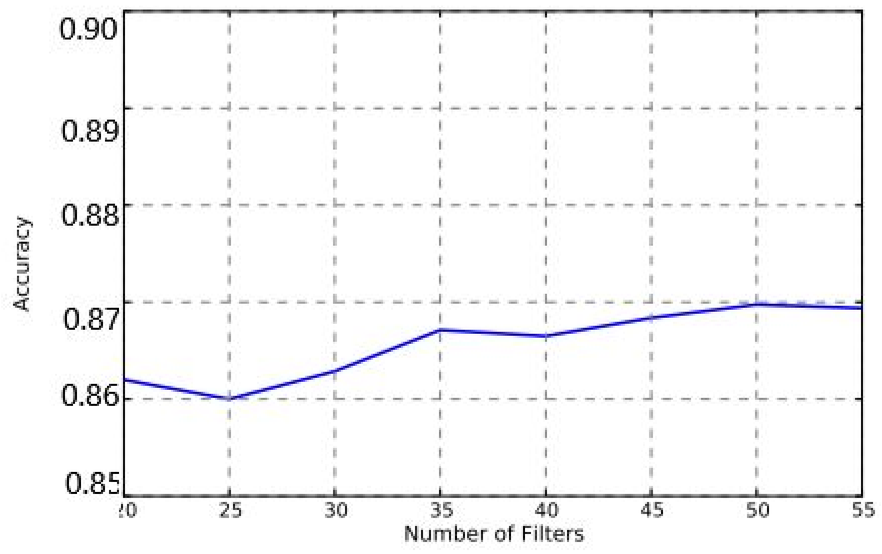
For each file in Train and Test data, the feature extraction pipeline is executed for each sound file. Following which, the Feature vector is cached and then stored as a .npy file within the same speaker's directory, which is named after the speaker.

For every classifier, the first part of the code essentially, retrieves the cached data of every audio file and adds it to the feature vector list, it also adds the Speaker label to a separate labels list (after converting the label to a numeral). This is done for both Train and Test dataset.

Trends in varying MFCC parameters -







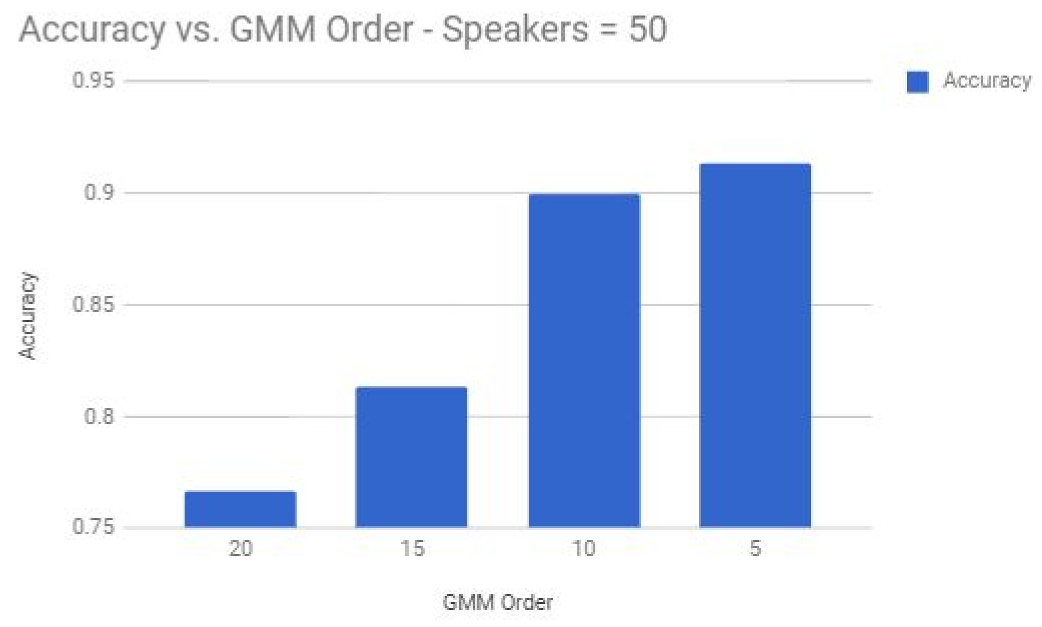
Gaussian Mixture Model -

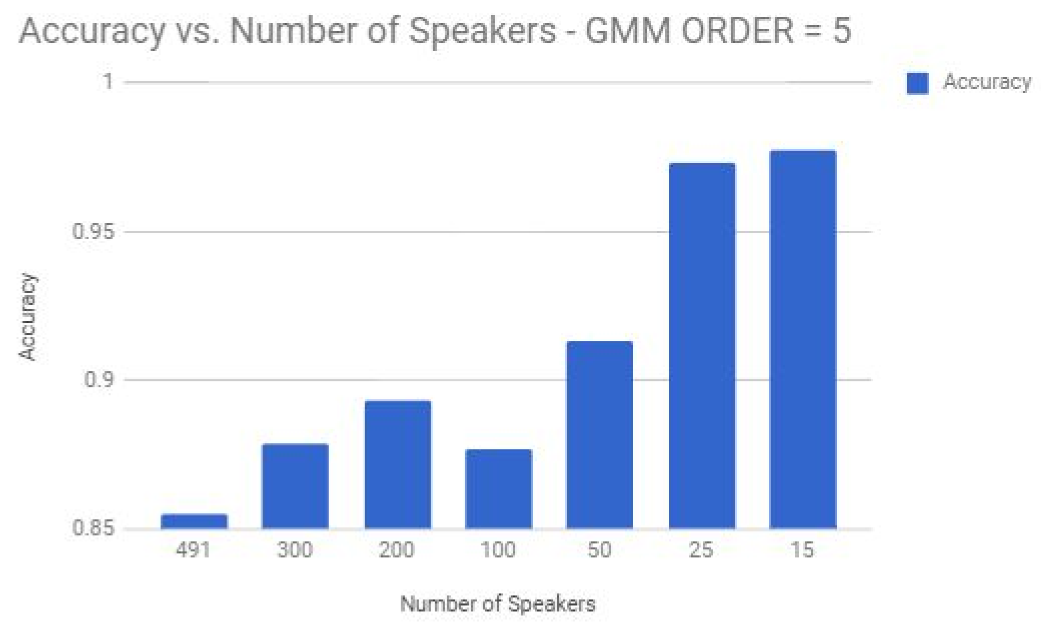
We Implemented a GMM with the following parameters that gave us an optimum result (For entire dataset) -

- MFCC frame length = 32ms, MFCC frame offset = 16ms
- MFCC filter banks = 15, number of cepstrums = 13

- Number of Gaussian Mixtures = 5
- Number of Speakers = 491 (entire dataset)

Accuracy for the above Parameters was 85.53%





Support Vector Machine -

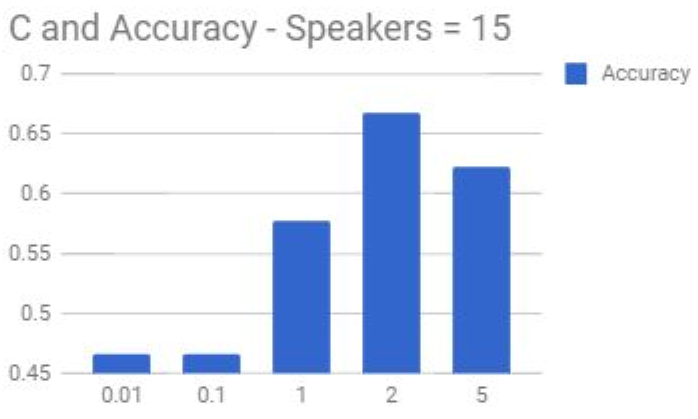
We have Varied the following parameters and taken results with each variation -

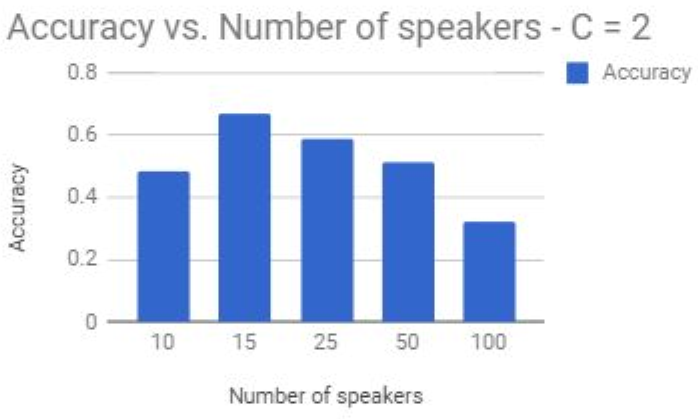
- Kernel - Linear, Poly, RBF
- C - 0.01, 0.1, 1, 2, 5, 10, 100, 500
- Number of Speakers - 10, 15, 25, 50, 100

The Best possible results we got was for **RBF Kernel, C = 100**

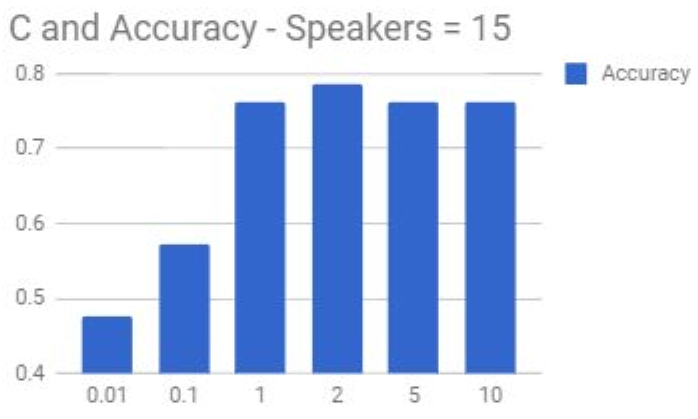
- **10 Speakers = 96.2% accuracy**
- **15 Speakers = 80.9% accuracy**
- **25 Speakers = 76.3% accuracy**

Linear

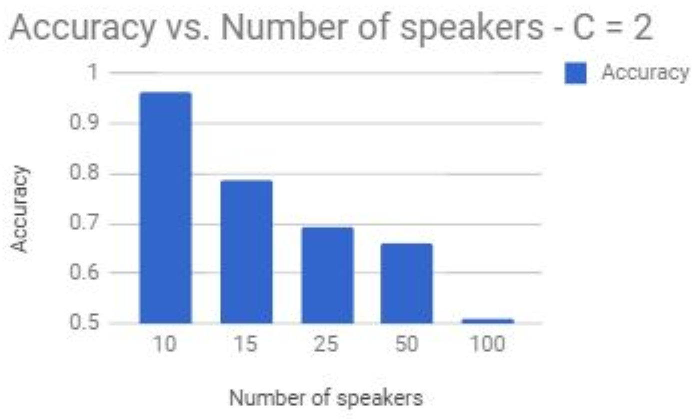




Poly

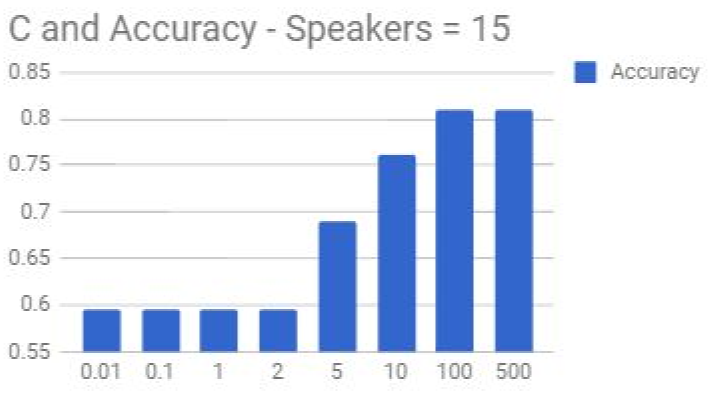


C and Accuracy - Speakers = 15
Accuracy
0.8
0.7
0.6
0.5
0.4
0.01
0.1
1
2
5
10

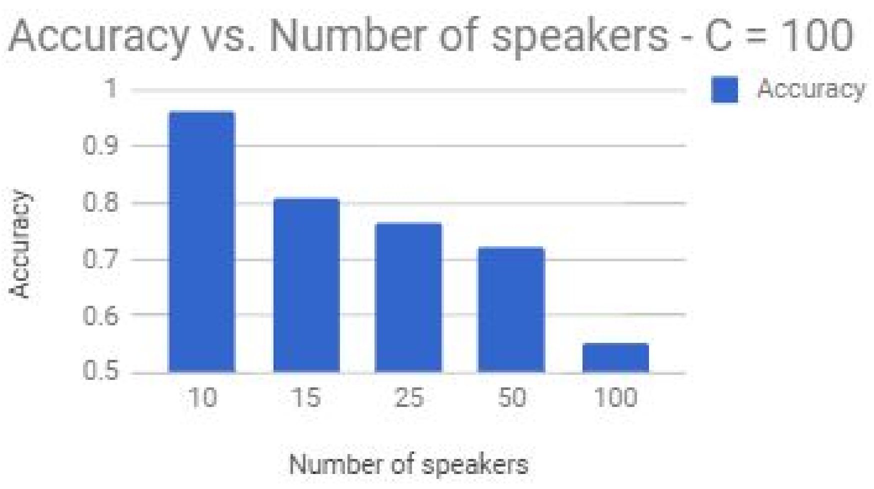


Accuracy vs. Number of speakers - C = 2
Accuracy
1
0.9
0.8
0.7
0.6
0.5
Accuracy
10
15
25
50
100
Number of speakers

RBF



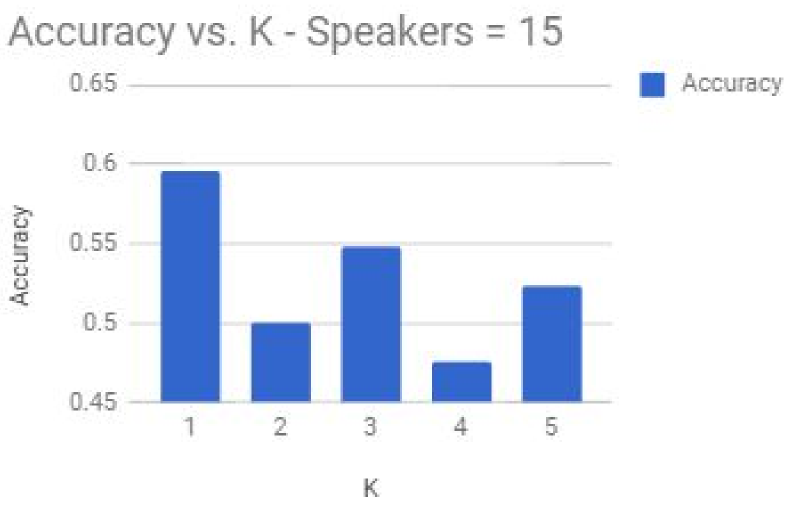
C and Accuracy - Speakers = 15
Accuracy
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.01
0.1
1
2
5
10
100
500

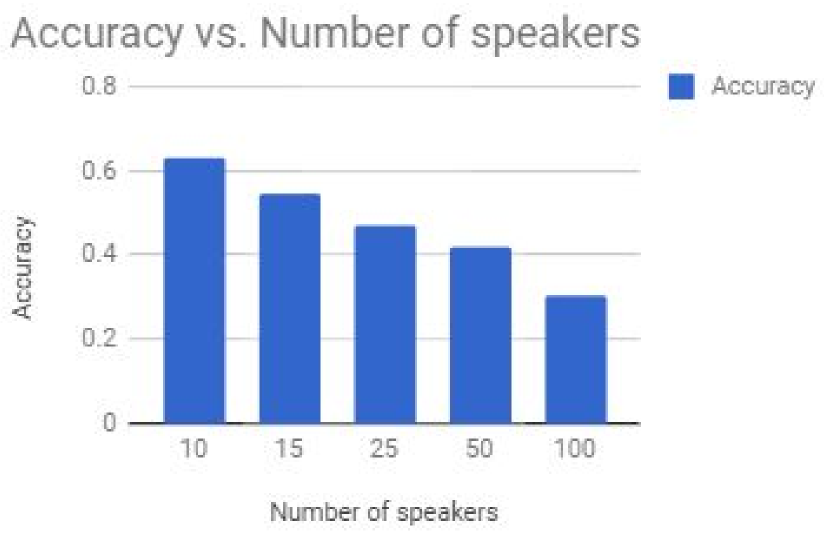


K Nearest Neighbours -

Using the KNN algorithm, the best possible accuracy we got was for the following -

K = 3, Number of Speakers = 10 - Accuracy = 63%

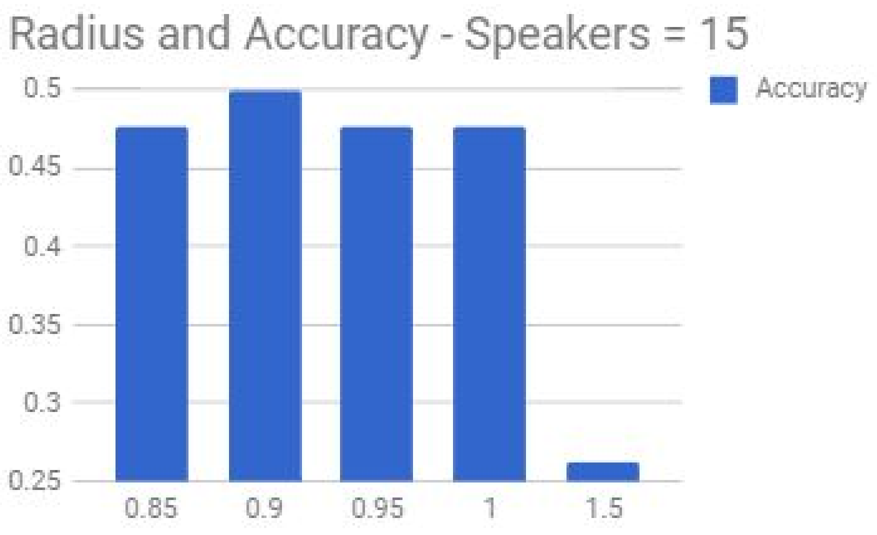


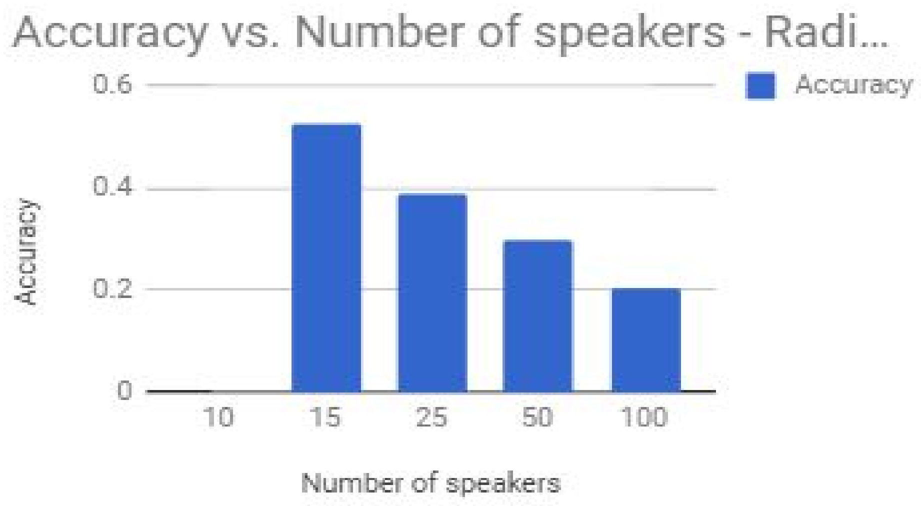


Radius Neighbourhood -

Best Accuracy with the following Parameters -

Radius = 0.97, Number of Speakers = 15 - Accuracy = 52.3%





Neural Network -

We had implemented a Neural Network with 2 Hidden layers.

Layer 1 -

- Dense Layer - 64 Filters
- Activation - tanh

Layer 2 -

- Dense Layer - Output = Number of classes
- Activation - tanh
- Optimizer function - SGD, sparse categorical

We were only partially able to implement this part of the project. We have coded the basic structure of the Network and were able to run it, but parameter selection has not yet been done.

The accuracies for this have been very low as a result.

References -

1. TIMIT dataset - https://goo.gl/6NBMLk
2. MFCC - https://en.wikipedia.org/wiki/Mel-frequency_cepstrum
3. Speaker identification and verification using Gaussian mixture speaker models, DA. Reynolds, 1995. http://www.sciencedirect.com/science/article/pii/016763939500009D

Libraries Used -

1. VAD - https://pypi.python.org/pypi/webrtcvad
2. MFCC - https://scikits.appspot.com/talkbox
3. Neural Networks - https://keras.io/
4. GMM, SVM, KNN - http://scikit-learn.org/
5. Utilities - http://www.numpy.org/