COEN 281 Term Project

Audio Mining - Predicting voice elicited features using data mining techniques.

Team 5

Arushi Gupta
Subrahmanya Pramod Nanduri
Aishwarya Gupte
Ruchika Garg
Surya Tamirisa

Instructor

Prof Ming Hwa Wang
Santa Clara University

Preface:

Audio/Voice is one of the critical areas of the advancement in the Data mining and intelligent systems in the present and the future. This project can predict the features of speech like Gender, Emotion, Speaker classification.

Acknowledgement

We would like to thank Dr. Ming-Hwa Wang for teaching us about the topics we used in this project.

Table of Contents

List of Figures:

List of Tables:

Abstract

Audio/Voice is one of the critical areas of the advancement in the Data mining and intelligent systems in the present and the future. For instance recent Google automated voice calling shows the amount of research going in this area. To stay on top of this, we want to deep dive into Audio Mining, where we would build a system that can predict the features of speech like Gender, Emotion, Speaker classification. The proposed system will extract features from the audio file in real time and perform analysis on these features for classification. The model will be an ensemble of data mining algorithms to reduce the number of false positives and improve the overall accuracy.

1. Introduction

1.1 Objective

The objective of the project is to process user's audio, to be used for work process automation. Given a speech input, our aim is to identify the speaker, detect speaker's gender, and analyze the speaker's emotion.

1.2 What is the problem

The common misconception with audio mining is that when a user says something, it is to be understood by the computer. However this is not the case and even humans misunderstand what someone is saying. This is due to various factors such as background noise, signal/noise ratio, speaker dependent and independent features, speech variability and language characteristics. The speech recognition technology is growing rapidly but has still not reached its full potential. Accuracy and reliability of the system is affected by unwanted inputs and low output as results. The first machine to recognise speech was developed in 1920 and a significant advancement has

been observed since then. We need to understand what speech recognition can be used for and what humans need it for and try to bridge the gap.

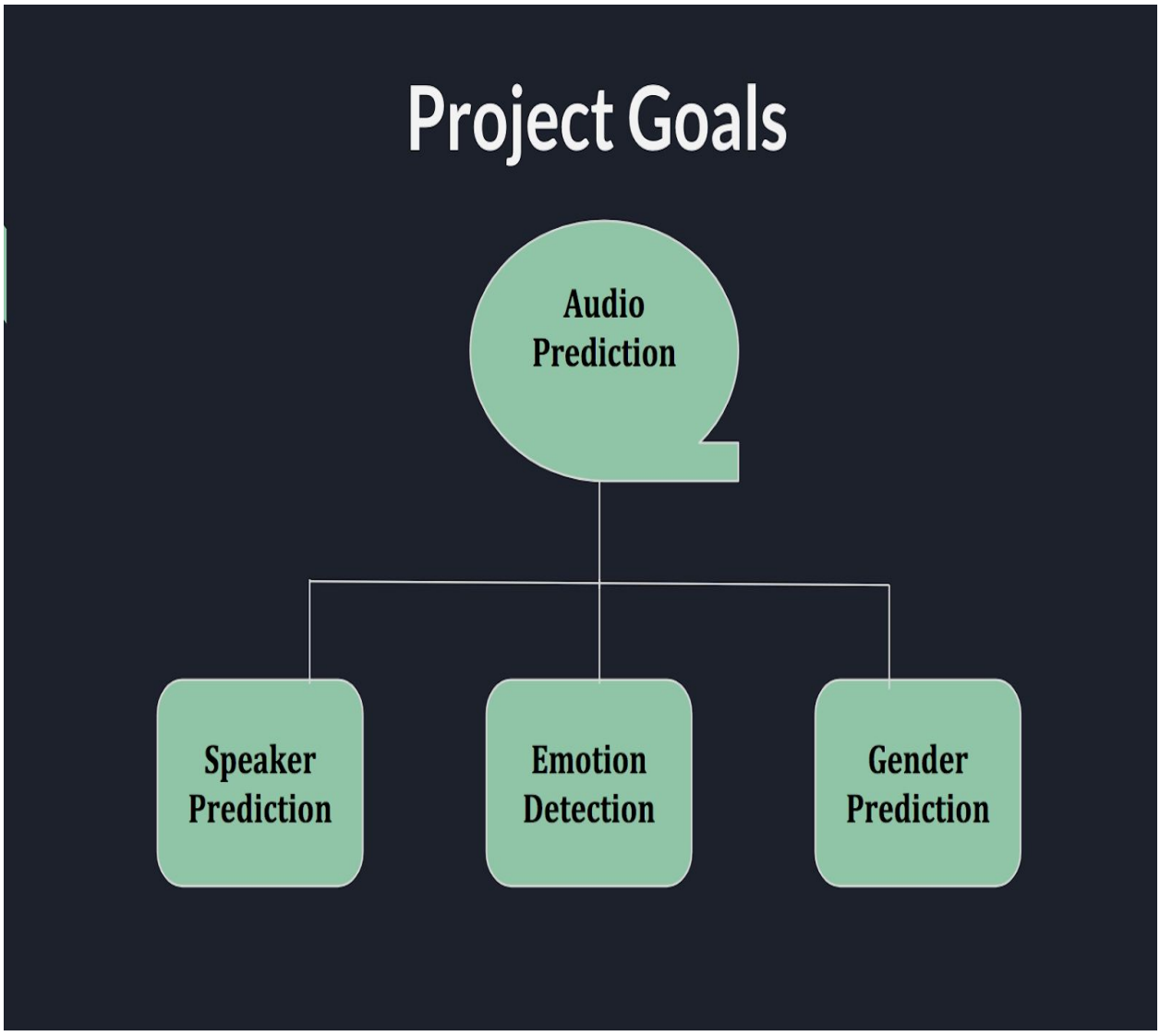


Figure 1: Project goals covering the Speaker, Emotion, Gender functionalities of the Audio signal.

1.3 Why this is a project related this class

Voice mining is one of the key project that involves lot of mining techniques and models. In this project, we are going to apply various mining models to extract useful information from the data of voice and will train the model to predict speaker classification, emotion prediction and gender prediction. Machine Learning has a connection with Data mining in terms of extracting an underlying representation and knowledge from the high dimensional data. In, data mining, we care about understanding the key insight of the given information which is related to our project in which we are interested in discovering various predictions through voice that can be applied to many industries. The approach we will take to model an accurate prediction system of gender, speaker, emotion will utilize multiple techniques in data mining and pattern recognition. First, by mining a large data set of voice and utilizing techniques in data mining and voice processing that are crucial for accurate feature extraction and predicting speaker accurately and emotions of the speaker and gender of the speaker.

1.4. Why other approach is no good

The most fundamental issue with the current approaches of audio mining is to choose the right type of mining algorithm and the accuracy of the algorithm.There are two main approaches to audio data mining . Firstly, Text-based indexing approach converts speech to text and then identifies words in a dictionary having several hundred thousand entries. If a word or name is not in the dictionary, the Large Vocabulary Continuous Speech Recognizers system will choose the most similar word it can find. Secondly, phoneme-based indexing approach analyzes and identifies sounds in a piece of audio content to create a phonetic-based index. It then uses a dictionary of several dozen phonemes to convert a user's search term to the correct phoneme string. Finally, the system looks for the search terms in the index. Both approaches have some drawbacks as LVCSR generates wrong results

when the word is not found in the dictionary and phonetic-based searches can result in more false matches than the text-based approach, particularly for short search terms, because many words sound alike or sound like parts of other words.

1.5. Why you think your approach is better

In our project, we are trying to build a single application that can perform all the functionalities of audio mining like gender recognition, speaker classification and emotion recognition. Instead of applying single model, we will try to apply different models and check for the highest accuracy model for particular functionality and will use that model for that particular functionality.

Our proposed approach has the following two steps:

1. Feature Extraction: It splits the input signal into short-term windows (frames) and computes a number of features for each frame. This process leads to a sequence of short-term feature vectors for the whole signal.In many cases, the signal is represented by statistics on the extracted short-term feature sequences and extracts a number of statistics (e.g. mean and standard deviation) over each short-term feature sequence.Features include like Energy,entropy of energy,MFCCs
2. Regression and Segmentation: Regression is important in audio analysis, e.g. in the context of speech emotion recognition, where the emotional state is not a discrete class but a real-valued measurement (e.g. arousal or valence). Segmentation is a very important processing stage for most of audio analysis applications. The goal is to split an uninterrupted audio signal into homogeneous segments. Segmentation can either be
    - Supervised: in that case some type of supervised knowledge is used to classify and segment the input signals. This is either achieved through applying a classifier in order to classify successive fix-sized segments to a set of predefined classes, or using HMM approach to achieve joint segmentation-classification.

- Unsupervised: a supervised model is not available and the detected segments are clustered (example: speaker diarization)

1.6 Area or scope of investigation

The web ,databases and other digitized information storehouses contain a large volume of audio content.For example newscasts,sporting events, telephone conversations,recording of meetings, webcasts etc. User wants to make the most of this material by searching and indexing the digitized audio content. In this project, we are trying to do audio mining and identify speaker classification as well as gender classification. Also, we are trying to apply data mining models to classify speaker emotions and we will compare different models to find out the most optimized algorithm and we are trying to improve the accuracy of prediction.

The continued ease of collecting and making available speech from real applications means that research can be focused on more real-world robustness issues that appear. Obtaining speech from a wide variety of handsets, channels and acoustic environments will allow examination of problem cases and development and application of new or improved techniques.

There are many other sources of speaker information in the speech signal that can be used. These include idiolect (word usage), prosodic measures and other long-term signal measures. The work will be aided by the increasing use of reliable speech recognition systems for speaker recognition R&D. High-level features not only offer the potential to improve accuracy, they may also help improve robustness since they should be less susceptible to channel effects.

2. Theoretical bases and literature review

2.1. Definition of the problem

The problem is analysing audio and mining features from the audio file to predict speaker, emotion and gender. Our goal is to generate most accurate model for prediction.

2.2. Theoretical Background Of The Problem

MFCCs are an accurate indicator of a speaker's gender. Mel-frequency cepstral coefficients together constitute a Mel-frequency cepstrum. MFCCs are derived in the following fashion -
Firstly, Fourier transform of a signal is taken, then powers of the spectrum are mapped above onto the mel scale, using triangular overlapping windows. The logs of the powers at each of the mel frequencies are taken. The discrete cosine transform of the list of mel log powers are taken as if it were a signal. The MFCCs are the amplitudes of the resulting spectrum.

2.3 Related research to solve the problem

The problem involves analysis of audio to predict speaker, emotion and gender. So research done as a part of audio mining project is focused on each of these sub-tasks.

Gender prediction - Gender has two classes - male and female. To predict gender, the insights gleaned from papers referred are as follows -
Gender was classified on basis of five modalities namely acoustic, linguistic, visual, thermal and physiological. Gender was predicted based on both single and combined modalities. It was found that highest accuracy using thermal regions - face, forehead, periorbital(eyes), cheeks, and nose was that from face. Considering physiological signals, the best results were obtained after combining all signals. The best results were obtained when vocal features

including MFCC were used. Linguistic features were not as useful as compared to the rest of the features. In case of visual features, the best results were obtained when all the features including face gestures, hand gestures and both, highest accuracy was obtained after all features were combined. It was concluded that vocal features were the most accurate indicator of gender. Moreover, gender can be recognised from audio from the features - Pitch, signals, Mel Frequency Cepstral Coefficients(MFCC).

Emotion Prediction - Emotions can be classified into three classes - positive, negative and neutral. Models are built for binary task of classifying sentiment into positive and negative classes and three way task of classifying into positive negative and neutral classes. The experiment was conducted with three types of models - unigram model, a feature based model and tree kernel based model. It was found that tree kernels outperform unigram and featured based models. Combining unigrams with feature based model outperforms combination of kernels with feature based model.

Speaker prediction - To predict speaker, features are derived from voice biometrics. To perform this, following insights were gleaned from the research papers referred.
The steps involved are training phase and matching phase. In the training phase, input speaker signal is pre-processed and its features are extracted. In matching phase, the speaker is identified by comparing the test speaker's voice features with existing models stored in database.

2.4 Advantages/Disadvantages of the Research

The research suggests that the use of MFCC over other Automatic Speech Recognition algorithms like, BFCC(Bark frequency Cepstral coefficient), LPCC(Mel Frequency Cepstral Coefficient), etc, will yield high accuracy rate.

By using three classes of emotion rather than two, we will be able to differentiate the ones in the border, as neutral. And for this classification we

will be using Kernel method, which outperforms, the n-gram method, and best system method.

The only and major disadvantage of MFCC is by the shape of the filter. Several filter coefficients might be negative because those might be the components of eigenvector of a covariance matrix, which means this may result in the output filter to be negative, thus fails to be converted to the log spectral domain.

2.5 Solution to solve this problem

To solve this typical Audio mining problem statement, we are going to integrate the SVM based techniques to classify various voice features.

2.6 Where your solution is different from others

We are going to try out SVM based technique instead of the traditional Vector distance model. We are also going to experiment other techniques like Random forest to see the performance.

2.7 Why your solution is better

For this classification we will be using SVM method, which out performs, the methods. We also build a whole framework where multiple voice features are extracted out of the single point system for uniform access. To understand the performance, our experiments with other models too will give comparitive analysis.

3. Hypothesis (or goals)

The goal of the project is to develop a user-friendly and efficient system for determining various speech characteristics. The system receives an input in the form of a digital audio file and analysis it frame by frame. Given an audio file, we want the system to be able to classify who is the speaker and what is the gender and emotion of the speaker. There are many algorithms to solve this problem and we select the best ones so that we can find the best features. Through a series of algorithms, the system evaluates the probabilities of each hypothesis.

3.1 Multiple hypothesis

Our model will reduce the number of false positives to improve the overall accuracy of the system. In other words, our model should have a low error rate.

3.2 Positive Hypothesis

Our goal is to take real time audio files from different geographical regions and create a system to get the best results using an ensemble of data mining algorithms.

4. Methodology

The project is divided into two phases.

1. Enrolment Phase or Training Phase: The enrolment or training phase is the initial phase where the input speaker signal is pre-processed and its features are extracted. Pre-processing is a form of cleansing to make it suitable to identify and extract characteristic features of the speaker signal. The process of feature extraction will enable the presentation of the speaker vocal characteristics to construct a model for that particular speaker.

2. Matching Phase: The matching phrase is responsible to identity the voice functionalities by comparing the test voice prints with the existing models stored in the database during the enrolment phase; the comparison of unique characteristic features is what defines 'matching'

So the input is passed through the enrollment phase and then into the matching phase which does the match with the existing samples.
We follow the sample process to extract multiple voice functionalities of Gender, voice and speaker.
We build a framework where we can extend to add further more functionalities into the system.

The figure which describes the above process is as shown below.

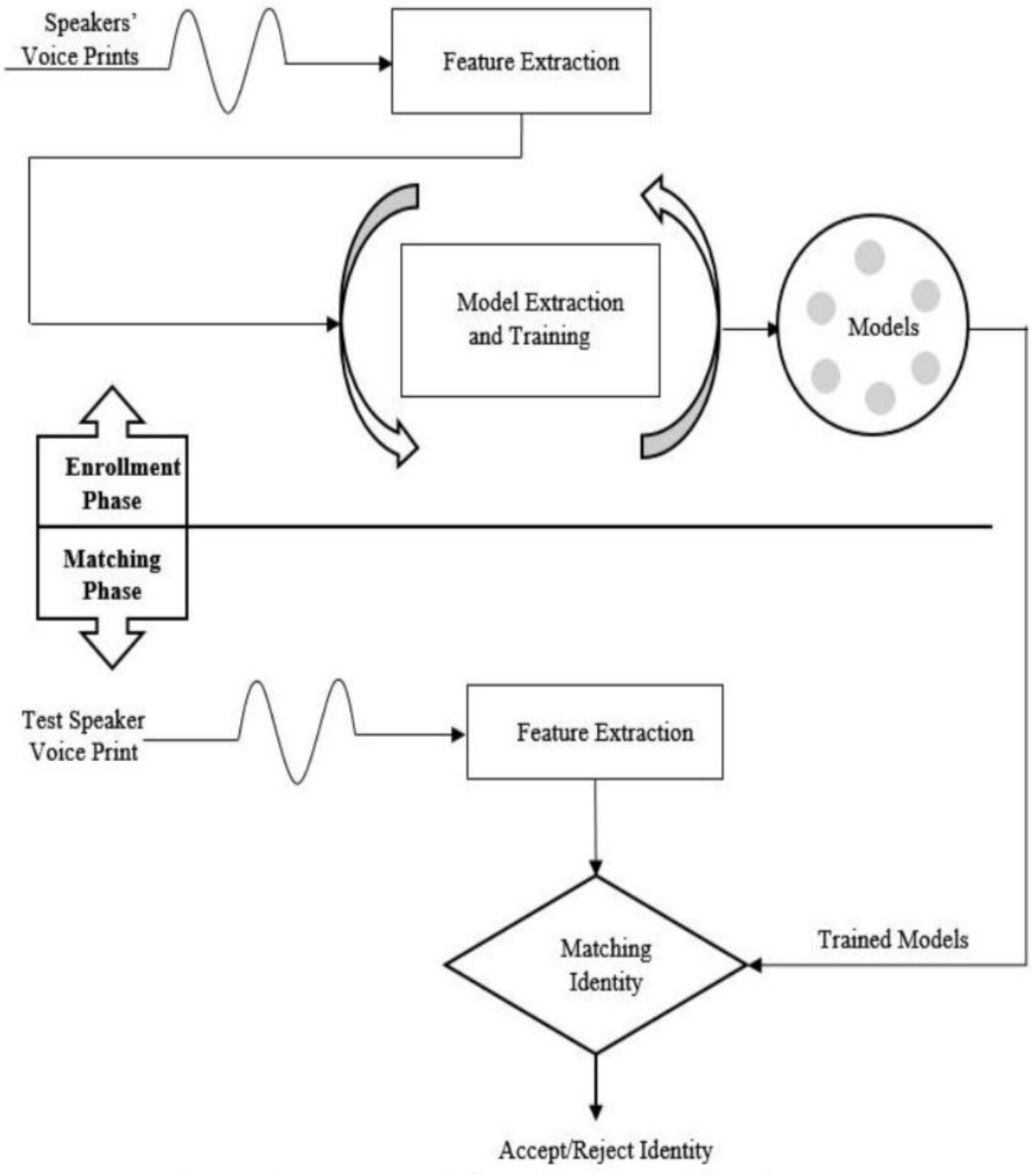


Figure 2: Shows two phases of Enrollment phase and matching phase

Deeper steps covering these two phases are as given below.

4.1 How to generate/collect input data

Test Input: To test the working of the system, input data can be generated by taking a Audio signal , which can be taken by either recording and feeding the voice signal or by importing the wav file into the system.

Training Dataset input:
Dataset that is used for training the system can be fed using a standard csv file of the features extracted or by feeding the audio signal. If the audio signal is used , we need an intermediate library which will convert the input signal to the format of the features. For the purpose of the demonstration, we are going to convert the speech signal to feature format by using pyAudio analysis open source library.

4.2 How to solve the problem

4.2.1 Algorithm design :
The simplified version of the problem we have at hand is to predict the voice related critical features like Gender, Speaker and Emotion .
To solve this, we follow the below process:

Input Parsing & Feature Extraction:
We take the training data collected from various public datasets available and extract the voice signals into various features as per the format we need.
To achieve this voice interpretation, we use an open-source library called pyAudioAnalysis to do the voice conversion to MFCC features.

PyAudioAnalysis:
PyAudioAnalysis is a Python library covering a wide range of audio analysis tasks. Through pyAudioAnalysis we can:

- Extract audio features and representations (e.g. mfccs, spectrogram, chromagram)
- Train, parameter tune and evaluate classifiers of audio segments
- Perform supervised segmentation
- Perform unsupervised segmentation
- Extract audio thumbnails
- Train and use audio regression models
- Apply dimensionality reduction to visualize audio data and content similarities

Features from Dataset:
Using PyAudioAnalysis, we extract the features from the audio signal . The features that we target to use are

Duration: length of signal
Meanfreq: mean frequency (in kHz)
SD: standard deviation of frequency
Median: median frequency (in kHz)
MFCC : Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFCC.

Each extracted feature data is tagged into respective label , for example for Gender we label the data as Male and Female, where are for Emotion we tag the labels are Angry, Happy and so on.

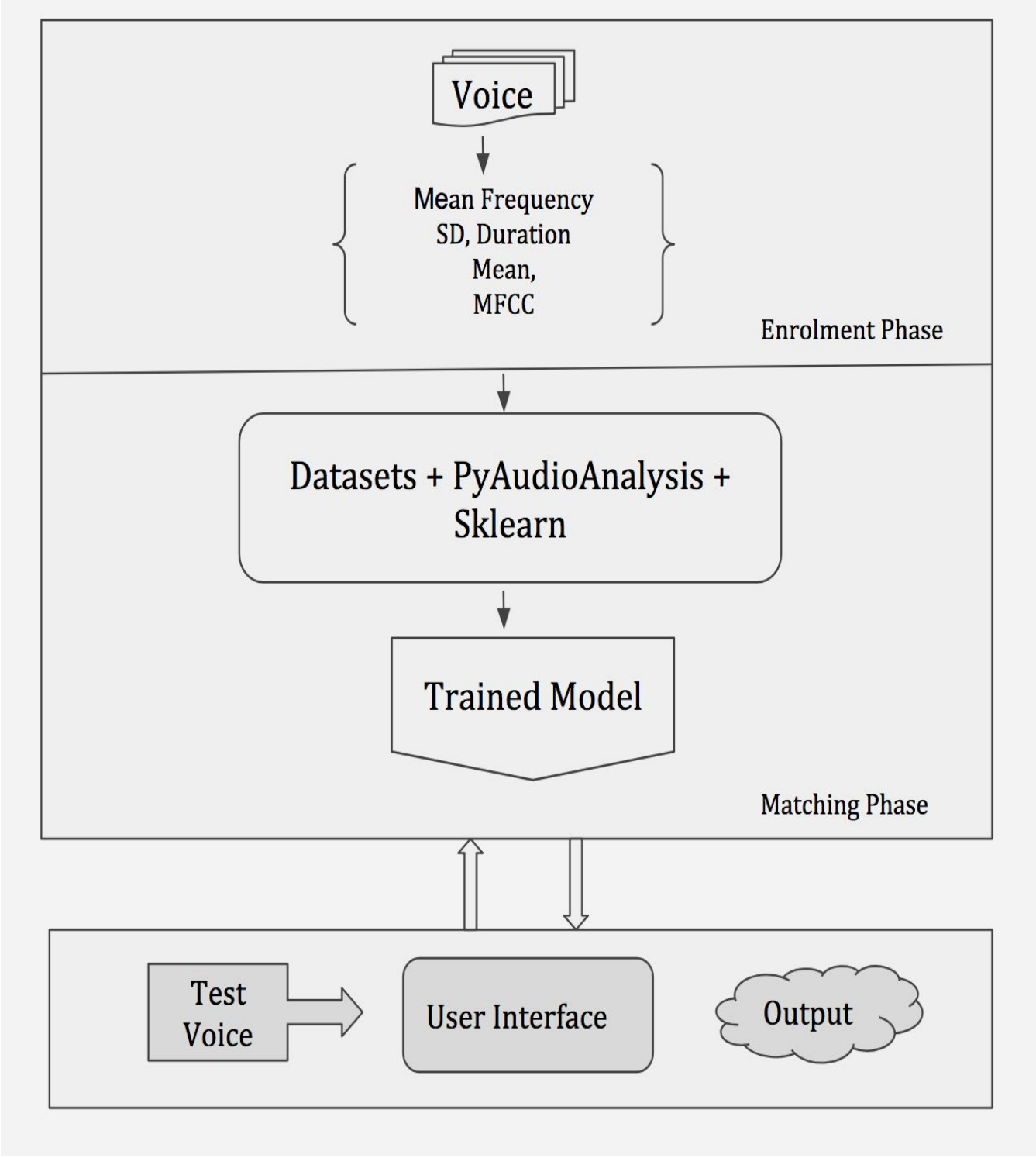


Figure 3: Flow Diagram for the whole project.

Algorithm & Actual Process:

- ➔ Input data set contains the voice signals that are parsed and MFCC features are extracted using the PyAudioAnalysis library and tagged as per the respective labels.
- ➔ Once we have the extracted feature set, we train the model using the SVM (Simple vector machine) algorithm to classify the voice features as per the desired label. During the training phase, each and every dataset item is given as the input to the Simple Vector machine algorithm for analysis of future data.
- ➔ We have the model trained with the training data based out of SVM and other algorithms, we get the model that can actually classify the future inputs.
- ➔ Once we have the model ready, we give the test input to check the accuracy of the model. Depending on the output, we tune various parameters of the system and consider various features for further dry runs to optimize the maximum efficiency.
- ➔ Now any test voice that needs to be checked with the model , is taken either by recording or by giving a wav file to the system.
- ➔ Our system converts the wav file into features and runs through the model and gives the output to the user.

Simple Vector Machine algorithm:

We plan to use SVM as our algorithm to train the model based out of supervised learning.

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.Suppose you are given plot of two label classes on graph as shown in image. Can you decide a separating line for the classes? This is what SVM does.

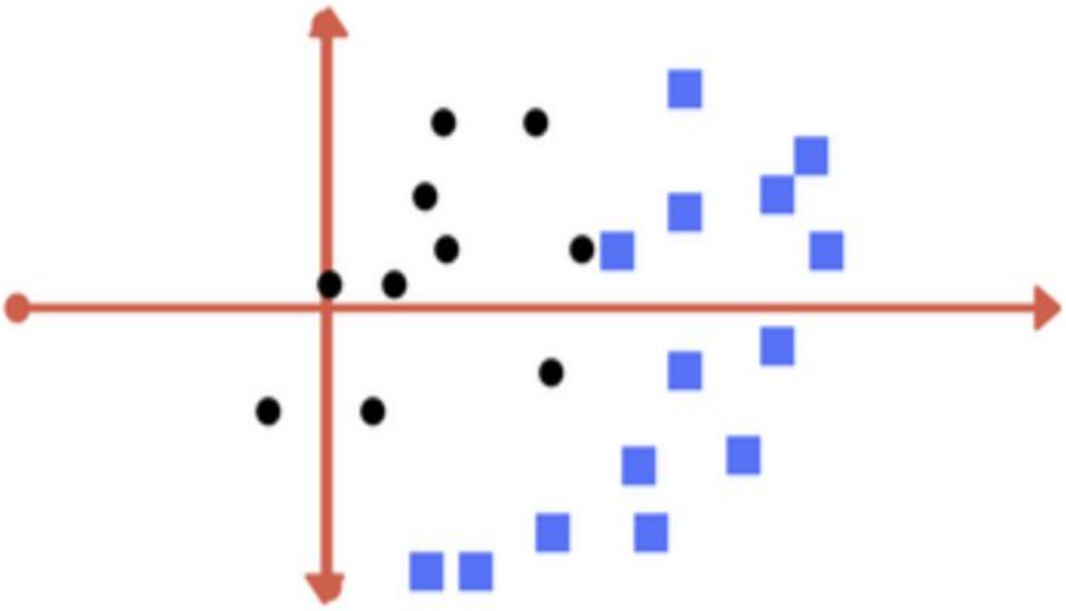


Fig 4: Input Label to SVM algorithm.

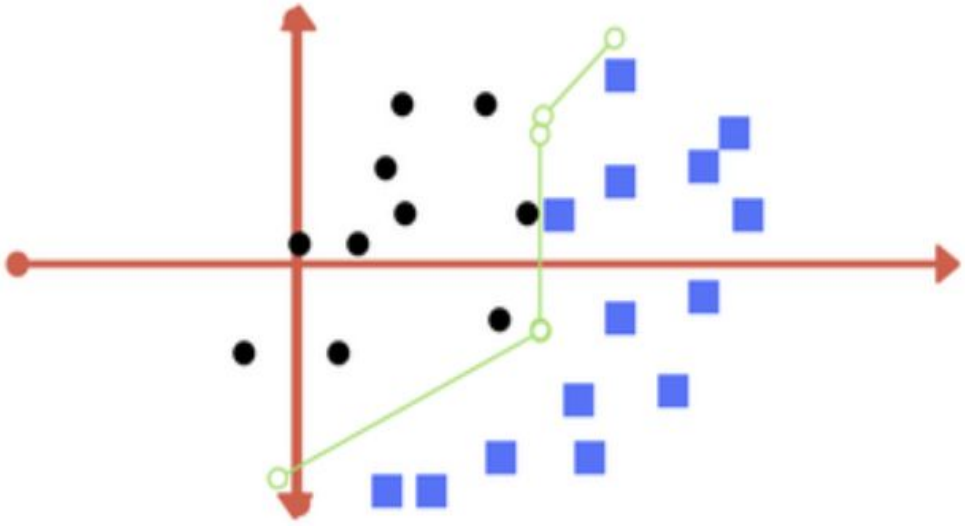


Fig 5: Output Hyperplane of SVM algorithm:

Since this is a best fit for use, we use SVM for training and classification of voice features.
We use scikit learn

4.2.2 Language used

Python 2.7+

4.2.3 Tools used

- PyAudioAnalysis library.
- Python utilities:
    - Matplotlib: Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
    - Scipy & Sklearn: Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language.Its features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
    - Hmmlearn
    - Simplejson: Simplejson exposes an API familiar to users of the standard library marshal and pickle modules. It is the externally maintained version of the json library contained in Python 2.6, but maintains compatibility with Python 2.5 and (currently) has significant performance advantages, even without using the optional C extension for speedups. simplejson is also supported on Python 3.3+.
    - Numpy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays

4.3 How to generate output :

To generate the output, we need to train the model and give an input file.
To train the model ,we follow the protocol as

python train_audiomining.py "dataset.path"

4.4 How to test against hypotheses

To generate the output, the protocol we follow is

python audiomining.py "input.wav"

Here audiominig.py is the python file that we write for the purpose of the demonstration. It takes basic input of input file path , defaulting to the current directory. Audiomining.py detects the voice features accordingly.

5. Implementation

5.1 Code

Gender Recognition:

Train Model for gender detection:

```
import os
import cPickle
import numpy as np
from scipy.io.wavfile import read
from sklearn.mixture import GMM
import python_speech_features as mfcc
from sklearn import preprocessing
import warnings
import csv
import pandas as pd

gender="female"
val=0


warnings.filterwarnings("ignore")

def get_MFCC(sr,audio):
    features = mfcc.mfcc(audio,sr, 0.025, 0.01, 13,appendEnergy
= False)
    features = preprocessing.scale(features)
    return features

#path to training data
source   =
"/Users/arushigupta148/Desktop/pygender/train_data/youtube/"+gen
der+"/"
#path to save trained model
dest     = "/Users/arushigupta148/Desktop/pygender/"
files    = [os.path.join(source,f) for f in os.listdir(source)
if
             f.endswith('.wav')]
features = np.asarray(())
```

```
for f in files:
    sr,audio = read(f)
    vector   = get_MFCC(sr,audio)
    if features.size == 0:
        features = vector
    else:
        features = np.vstack((features, vector))

gmm = GMM(n_components = 8, n_iter = 200,
covariance_type='diag',
        n_init = 3)
gmm.fit(features)
picklefile = f.split("/")[-2].split(".wav")[0]+".gmm"


#Create csv file

csvfile="/Users/arushigupta148/Desktop/"+gender+".csv"
with open(csvfile, "w") as output:
    writer = csv.writer(output, lineterminator='\n')
    writer.writerows(features)


#Add new column to csv file
df = pd.read_csv(gender+".csv")
df[val] = val
df.to_csv(gender+".csv")


# model saved as male.gmm
cPickle.dump(gmm,open(dest + picklefile,'w'))
print 'modeling completed for gender:',picklefile
```

Test Model for gender detection:

```
import os
import cPickle
import numpy as np
from scipy.io.wavfile import read
import python_speech_features as mfcc
from sklearn import preprocessing
import warnings
warnings.filterwarnings("ignore")
def get_MFCC(sr,audio):
```

```
    features = mfcc.mfcc(audio,sr, 0.025, 0.01, 13,appendEnergy
= False)
    feat     = np.asarray(())
    for i in range(features.shape[0]):
        temp = features[i,:]
        if np.isnan(np.min(temp)):
            continue
        else:
            if feat.size == 0:
                feat = temp
            else:
                feat = np.vstack((feat, temp))
    features = feat;
    features = preprocessing.scale(features)
    return features

#path to test data
sourcepath =
"/Users/arushigupta148/Desktop/pygender/test_data/AudioSet/femal
e_clips/"
#path to saved models
modelpath  = "/Users/arushigupta148/Desktop/pygender/"

gmm_files = [os.path.join(modelpath,fname) for fname in
             os.listdir(modelpath) if fname.endswith('.gmm')]

models    = [cPickle.load(open(fname,'r')) for fname in
gmm_files]
genders   = [fname.split("/")[-1].split(".gmm")[0] for fname
             in gmm_files]
files     = [os.path.join(sourcepath,f) for f in
os.listdir(sourcepath)
             if f.endswith(".wav")]

for f in files:
    print f.split("/")[-1]
    sr, audio  = read(f)
    features   = get_MFCC(sr,audio)
    scores     = None
    log_likelihood = np.zeros(len(models))
    for i in range(len(models)):
        gmm    = models[i]         #checking with each model one
by one
        scores = np.array(gmm.score(features))
        log_likelihood[i] = scores.sum()
```

```
    winner = np.argmax(log_likelihood)
    print "\tdetected as - ", genders[winner],"\n\tscores:female
",log_likelihood[0],",male ", log_likelihood[1],"\n"
```

Logistic Regression:

Regression can be rather important in audio analysis. The input files are taken in csv format. The data from the female and male csv files are read and their first columns are dropped to be merged into one file.

```
df = pd.read_csv("/Users/arushigupta148/Desktop/female.csv",
                 names = ["f1", "f2", "f3","f4", "f5", "f6","f7", "f8", "f9","f10", "f11", "f12","f13"])
df=df.drop(df.columns[0],axis=1)

df1 = pd.read_csv("/Users/arushigupta148/Desktop/male.csv",names = ["f1", "f2", "f3","f4", "f5", "f6","f7", "f8", "f9","f10", "f11", "f12","f13"])
df1=df1.drop(df1.columns[0],axis=1)
```

Figure 6 : Reading male and female data

The two csv files are merged together into a new csv file. The variable X stores the features and the y variable has the labels for the data files.

```
frames = [df, df1]
result = pd.concat(frames)
result.to_csv("result.csv")

df3 = pd.read_csv("result.csv")
df3=df3.drop(df3.columns[0],axis=1)
df3.head()

print("Number of samples:",df.shape[0])
X, y = df3.iloc[:, :-1].values, df3.iloc[:, -1].values
```

Figure 7 : Merge csv files and assign labels

Extract the training and testing data, the testing size here is taken as 1 percent of the entire file size.

```
gender_encoder = LabelEncoder()
y = gender_encoder.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.01)
```

Figure 8: Emotions determined based on arousal and valence values

Emotion Detection:

The values for valence and arousal are calculated and accordingly the emotion of the speaker is determined. Four emotions are classified based on the values of arousal and valence.

```
if arr[1]>0 and arr[3]>0:
    print "happy"
elif arr[1]>0 and arr[3]<0:
    print "angry"
elif arr[1]<0 and arr[3]>0:
    print "calm"
else:
    print "sad"
```

Figure 9: Emotions determined based on arousal and valence values

Speaker Recognition:

```
#!/usr/bin/env python2
# -*- coding: UTF-8 -*-
# File: speaker-recognition.py
# Date: Sun Feb 22 22:36:46 2015 +0800
```

```
# Author: Yuxin Wu <ppwwyyxxc@gmail.com>

import argparse
import sys
import glob
import os
import itertools
import scipy.io.wavfile as wavfile

sys.path.append(os.path.join(
    os.path.dirname(os.path.realpath(__file__)),
    'gui'))
from gui.interface import ModelInterface
from gui.utils import read_wav
from filters.silence import remove_silence

def get_args():
    desc = "Speaker Recognition Command Line Tool"
    epilog = """
Wav files in each input directory will be labeled as the
basename of the directory.
Note that wildcard inputs should be *quoted*, and they will be
sent to glob.glob module.

Examples:
    Train (enroll a list of person named person*, and mary, with
wav files under corresponding directories):
    ./speaker-recognition.py -t enroll -i "./bob/ ./mary/
./person*" -m model.out

    Predict (predict the speaker of all wav files):
    ./speaker-recognition.py -t predict -i "./*.wav" -m
model.out
"""
    parser =
argparse.ArgumentParser(description=desc,epilog=epilog,

formatter_class=argparse.RawDescriptionHelpFormatter)

    parser.add_argument('-t', '--task',
                        help='Task to do. Either "enroll" or
"predict"',
                        required=True)

    parser.add_argument('-i', '--input',
```

```
                        help='Input Files(to predict) or
Directories(to enroll)',
                        required=True)

    parser.add_argument('-m', '--model',
                        help='Model file to save(in enroll) or
use(in predict)',
                        required=True)

    ret = parser.parse_args()
    return ret

def task_enroll(input_dirs, output_model):
    m = ModelInterface()
    input_dirs = [os.path.expanduser(k) for k in
input_dirs.strip().split()]
    dirs = itertools.chain(*(glob.glob(d) for d in input_dirs))
    dirs = [d for d in dirs if os.path.isdir(d)]
    files = []
    if len(dirs) == 0:
        #print "No valid directory found!"
        sys.exit(1)
    for d in dirs:
        label = os.path.basename(d.rstrip('/'))

        wavs = glob.glob(d + '/*.wav')
        print (wavs)
        if len(wavs) == 0:
            print ("No wav file found in {0}".format(d))
            continue
        print ("Label {0} has files {1}".format(label,
','.join(wavs)))
        for wav in wavs:
            fs, signal = read_wav(wav)
            m.enroll(label, fs, signal)

    m.train()
    m.dump(output_model)

def task_predict(input_files, input_model):
    m = ModelInterface.load(input_model)
    for f in glob.glob(os.path.expanduser(input_files)):
        fs, signal = read_wav(f)
        label = m.predict(fs, signal)
        print (f, '->', label)
```

```
if __name__ == '__main__':
    global args
    args = get_args()

    task = args.task
    if task == 'enroll':
        task_enroll(args.input, args.model)
    elif task == 'predict':
        task_predict(args.input, args.model)
```

5.2 Design Document and Flowchart

The project is designed in three modules. Voice for gender detection was processed by GMM and logistic regression. As we have to differentiate between male and female, we chose Logistic regression, as dependent variable is dichotomous The accuracy GMM was 86% for both male and female The logistic regression model achieves an accuracy of 72% on the training set and 71% on the testing set. This is clearly an improvement over the baseline algorithms. Speaker identification uses the GMM algorithm and Emotion detection is done by SVM.

First the data is trained with a training set. Based on this, the algorithm/program will yield results for the test dataset.

Emotion Detection in detail:

From the above figure, it is shown how emotions are classified in SVM algorithm. Out of which, we have chosen only 4 emotions, namely, happy, angry, sad and calm.

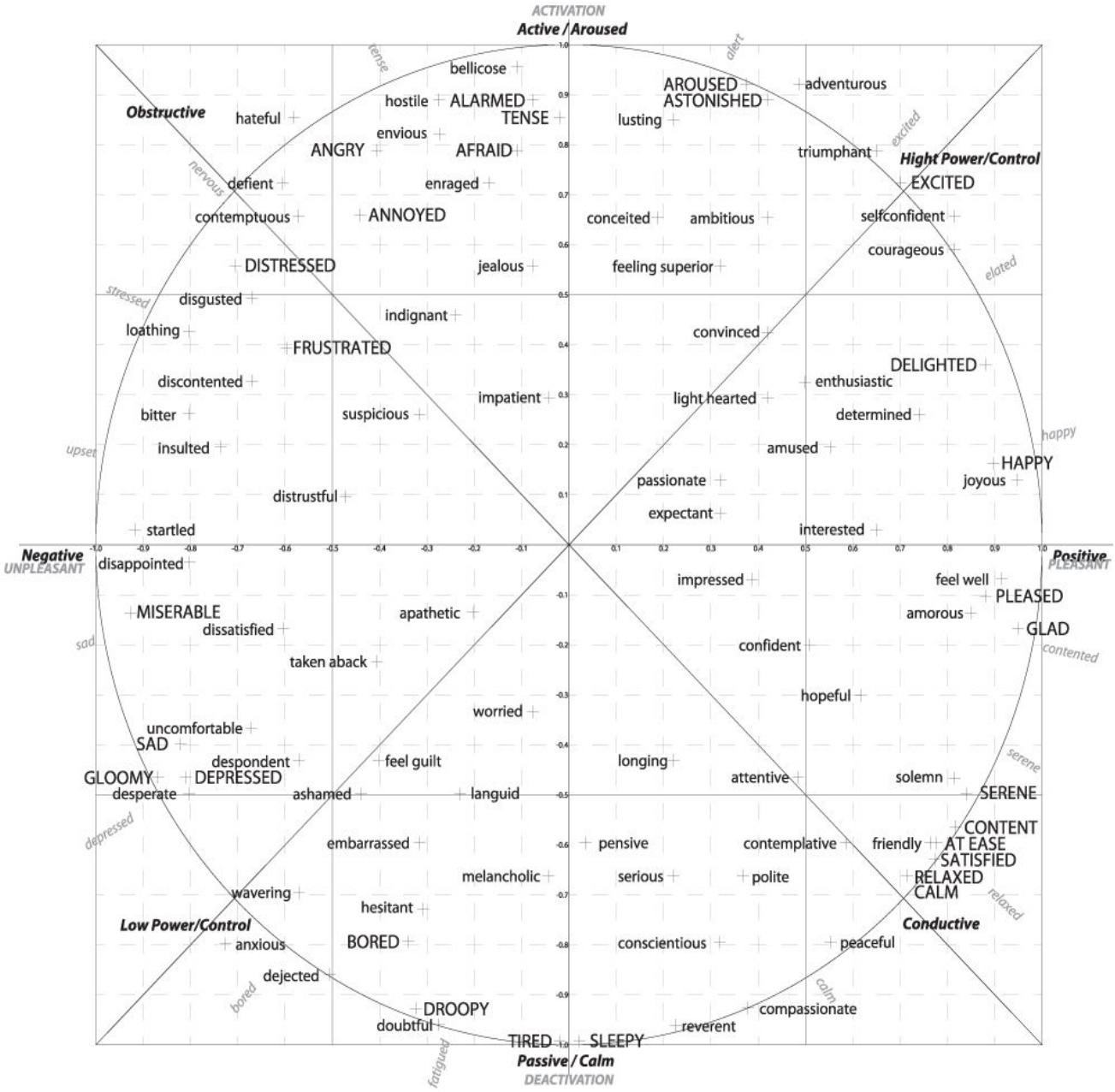


Fig 10: boundaries of emotion detection through SVM

If you observe, we have chosen the elements of differentiation, from four different quadrants. From this we have now clearly divided our graph as below.

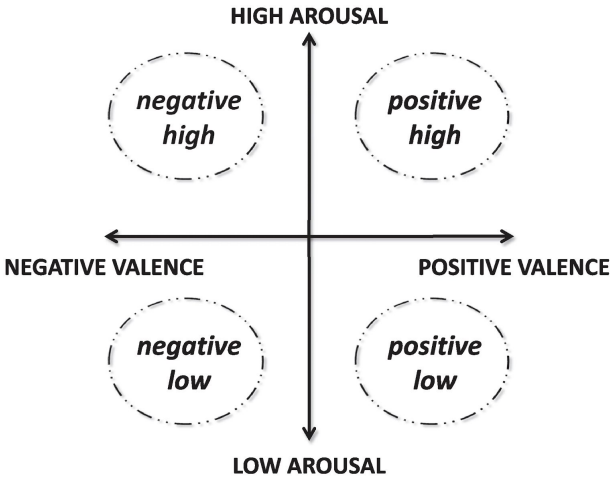


Fig 11: Valence/Arousal graph.

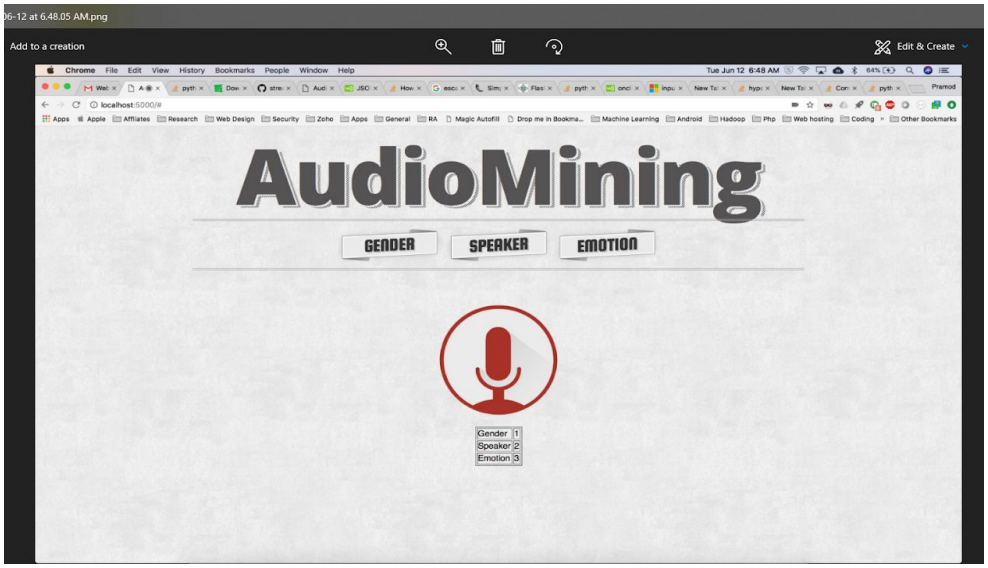
6. Data Analysis and Discussion

6.1 Output Generation

The input dataset containing voice .wav format files, which are first used to train the data. After training, the algorithm is tested with test data to check the results.

6.2 Output Analysis

Front end Screen



Gender Identification Output:

Logistic regression used for gender detection yields 80.6% accuracy.

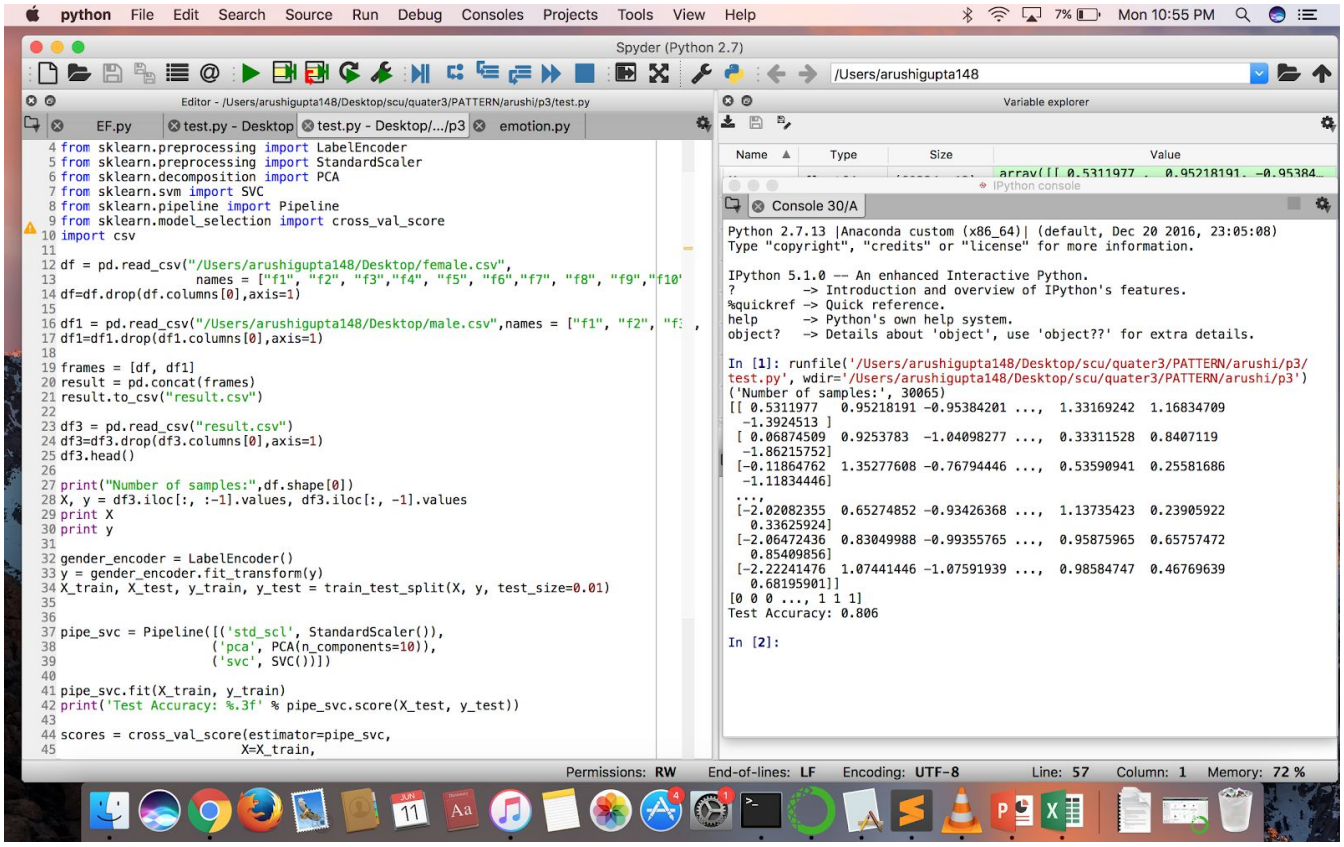


Figure 12: Output for Logistic Regression for Gender Detection

Emotion Identification:

The below figure shows the output for Emotion detection through SVM.

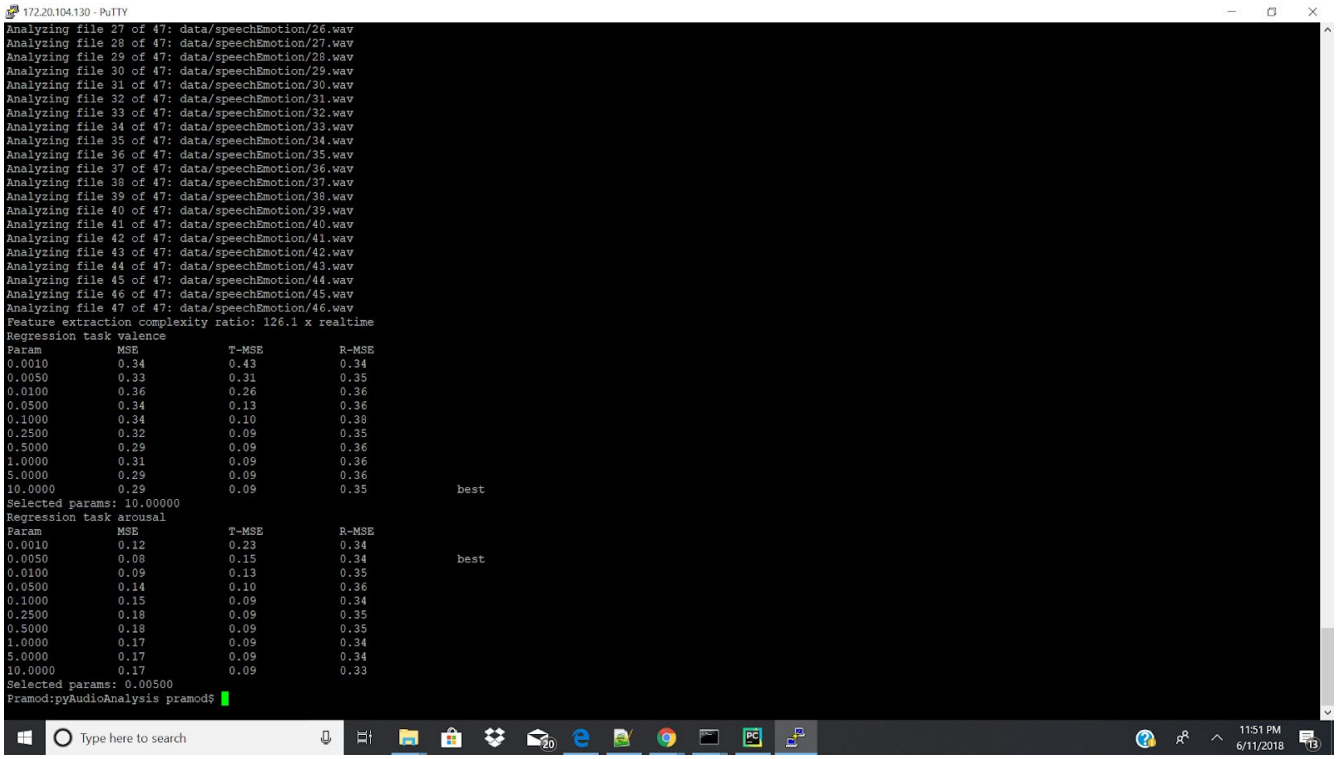
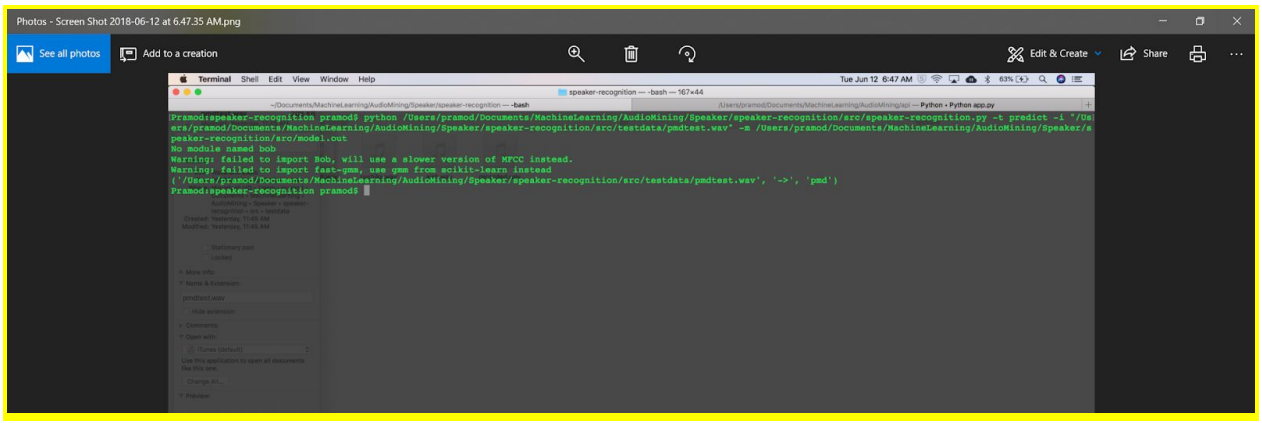


Fig 13 : Emotion detection through SVM.

The first column represents the resulting MSE for the respective SVM C param. The second comuns shows the MSE achieved on the training dataset (this is to provide a level of "overfitting"), while the last column shows the "baseline" MSE, i.e. the MSE achieved when the unknown variable is always set equal to the average value of the training set.

Speaker Recognition Output:



6.3 Compare Output against hypothesis

Hypothesis: Our goal is to take real time audio files from different geographical regions and create a system to get the best results using an ensemble of data mining algorithms.
We have created an ensemble of logistic regression and gaussian mixture model for gender detection. The program takes an audio wav file and determines the gender and emotion of the speaker. We also populated the dataset with our voice samples for training the speaker recognition dataset and testing the dataset with another wav file to determine the speaker.

6.4 Abnormal Case explanation

With less samples in the training sets, these algorithms will have a very low accuracy. The accuracy is pretty high, like mentioned in the beginning of the chapter, provided, the train data has a decent number of samples.

6.5 Discussion

Although Logistic regression is one of the important algorithms for gender classification, GMM out performs it, as it yields higher accuracy.

7. Conclusions and Recommendations

7.1 Summary and Conclusions

In this project, we have implemented 3 different data mining techniques, gender detection, speaker prediction and emotion analysis. We have combined these three data mining techniques, to build an application, which reads, audio, and implements the above. Gender detection uses GMM algorithm, speaker prediction uses GMM algorithm, and emotion analysis uses SVM algorithm.

7.2 Recommendation for future studies

This project can further be designed with higher accuracy, as an automation system like google's duplex, which depends on all the three parameters which we have worked upon. The boundaries of emotion detection can be redesigned, and can always be explored.

8. Bibliography

[1] Salekin, Asif, et al. "Distant Emotion Recognition." *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1.3 (2017): 96.

[2] Li, Ying, Jose D. Contreras, and Luis J. Salazar. "Predicting voice elicited emotions." *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015.

[3] Meo, Rosa, and Emilio Sulis. "Processing affect in social media: a comparison of methods to distinguish emotions in tweets." *ACM Transactions on Internet Technology (TOIT)*17.1 (2017): 7.

[4] Agarwal, Apoorv, et al. "Sentiment analysis of twitter data." *Proceedings of the workshop on languages in social media*. Association for Computational Linguistics, 2011

[5] Li, Panpan, et al. "Short Text Emotion Analysis Based on Recurrent Neural Network." *Proceedings of the 6th International Conference on Information Engineering*. ACM, 2017.

[6] Abouelenien, Mohamed, et al. "Gender-based multimodal deception detection." *Proceedings of the Symposium on Applied Computing*. ACM, 2017.

[7] Abouelenien, Mohamed, et al. "Multimodal gender detection." *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. ACM, 2017

[8] Yoo, In-Chul, Hyeontaek Lim, and Dongsuk Yook. "Formant-based robust voice activity detection." *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 23.12 (2015): 2238-2245.

[9] Ramdinmawii, Esther, and V. K. Mittal. "Gender identification from speech signal by examining the speech production characteristics." *Signal Processing and Communication (ICSC), 2016 International Conference on*. IEEE, 2016

[10] Campos, Victor de Abreu, and Daniel Carlos Guimarães Pedronette. "Effective Speaker Retrieval and Recognition through Vector Quantization and Unsupervised Distance Learning." *Proceedings of the 1st International Workshop on Multimedia Analysis and Retrieval for Multimodal Interaction*. ACM, 2016.

[11] Tirumala, Sreenivas Sremath, and Seyed Reza Shahamiri. "A review on Deep Learning approaches in Speaker Identification." *Proceedings of the 8th international conference on signal processing systems*. ACM, 2016.

9. Appendices

9.1 Program Source Code with Documentation

Gender Recognition:

Train Model for gender detection:

```
import os
import cPickle
import numpy as np
from scipy.io.wavfile import read
from sklearn.mixture import GMM
import python_speech_features as mfcc
from sklearn import preprocessing
import warnings
import csv
import pandas as pd

gender="female"
val=0


warnings.filterwarnings("ignore")

def get_MFCC(sr,audio):
    features = mfcc.mfcc(audio,sr, 0.025, 0.01, 13,appendEnergy
= False)
    features = preprocessing.scale(features)
    return features

#path to training data
source   =
"/Users/arushigupta148/Desktop/pygender/train_data/youtube/"+gen
der+"/"
#path to save trained model
dest     = "/Users/arushigupta148/Desktop/pygender/"
files    = [os.path.join(source,f) for f in os.listdir(source)
if
             f.endswith('.wav')]
features = np.asarray(())
```

```
for f in files:
    sr,audio = read(f)
    vector   = get_MFCC(sr,audio)
    if features.size == 0:
        features = vector
    else:
        features = np.vstack((features, vector))

gmm = GMM(n_components = 8, n_iter = 200,
covariance_type='diag',
        n_init = 3)
gmm.fit(features)
picklefile = f.split("/")[-2].split(".wav")[0]+".gmm"


#Create csv file

csvfile="/Users/arushigupta148/Desktop/"+gender+".csv"
with open(csvfile, "w") as output:
    writer = csv.writer(output, lineterminator='\n')
    writer.writerows(features)


#Add new column to csv file
df = pd.read_csv(gender+".csv")
df[val] = val
df.to_csv(gender+".csv")


# model saved as male.gmm
cPickle.dump(gmm,open(dest + picklefile,'w'))
print 'modeling completed for gender:',picklefile
```

Test Model for gender detection:

```
import os
import cPickle
import numpy as np
from scipy.io.wavfile import read
import python_speech_features as mfcc
from sklearn import preprocessing
import warnings
warnings.filterwarnings("ignore")
```

```
def get_MFCC(sr,audio):
    features = mfcc.mfcc(audio,sr, 0.025, 0.01, 13,appendEnergy
= False)
    feat     = np.asarray(())
    for i in range(features.shape[0]):
        temp = features[i,:]
        if np.isnan(np.min(temp)):
            continue
        else:
            if feat.size == 0:
                feat = temp
            else:
                feat = np.vstack((feat, temp))
    features = feat;
    features = preprocessing.scale(features)
    return features

#path to test data
sourcepath =
"/Users/arushigupta148/Desktop/pygender/test_data/AudioSet/femal
e_clips/"
#path to saved models
modelpath  = "/Users/arushigupta148/Desktop/pygender/"

gmm_files = [os.path.join(modelpath,fname) for fname in
             os.listdir(modelpath) if fname.endswith('.gmm')]

models    = [cPickle.load(open(fname,'r')) for fname in
gmm_files]
genders   = [fname.split("/")[-1].split(".gmm")[0] for fname
             in gmm_files]
files     = [os.path.join(sourcepath,f) for f in
os.listdir(sourcepath)
             if f.endswith(".wav")]

for f in files:
    print f.split("/")[-1]
    sr, audio  = read(f)
    features   = get_MFCC(sr,audio)
    scores     = None
    log_likelihood = np.zeros(len(models))
    for i in range(len(models)):
        gmm    = models[i]         #checking with each model one
by one
        scores = np.array(gmm.score(features))
```

```
        log_likelihood[i] = scores.sum()
    winner = np.argmax(log_likelihood)
    print "\tdetected as - ", genders[winner],"\n\tscores:female
",log_likelihood[0],",male ", log_likelihood[1],"\n"
```

Speaker Recognition:

```
#!/usr/bin/env python2
# -*- coding: UTF-8 -*-
# File: speaker-recognition.py
# Date: Sun Feb 22 22:36:46 2015 +0800
# Author: Yuxin Wu <ppwwyyxxc@gmail.com>

import argparse
import sys
import glob
import os
import itertools
import scipy.io.wavfile as wavfile

sys.path.append(os.path.join(
    os.path.dirname(os.path.realpath(__file__)),
    'gui'))
from gui.interface import ModelInterface
from gui.utils import read_wav
from filters.silence import remove_silence

def get_args():
    desc = "Speaker Recognition Command Line Tool"
    epilog = """
Wav files in each input directory will be labeled as the
basename of the directory.
Note that wildcard inputs should be *quoted*, and they will be
sent to glob.glob module.

Examples:
    Train (enroll a list of person named person*, and mary, with
wav files under corresponding directories):
    ./speaker-recognition.py -t enroll -i "./bob/ ./mary/
./person*" -m model.out

    Predict (predict the speaker of all wav files):
    ./speaker-recognition.py -t predict -i "./*.wav" -m
model.out
```

```
"""
    parser =
argparse.ArgumentParser(description=desc,epilog=epilog,

formatter_class=argparse.RawDescriptionHelpFormatter)

    parser.add_argument('-t', '--task',
                        help='Task to do. Either "enroll" or
"predict"',
                        required=True)

    parser.add_argument('-i', '--input',
                        help='Input Files(to predict) or
Directories(to enroll)',
                        required=True)

    parser.add_argument('-m', '--model',
                        help='Model file to save(in enroll) or
use(in predict)',
                        required=True)

    ret = parser.parse_args()
    return ret

def task_enroll(input_dirs, output_model):
    m = ModelInterface()
    input_dirs = [os.path.expanduser(k) for k in
input_dirs.strip().split()]
    dirs = itertools.chain(*(glob.glob(d) for d in input_dirs))
    dirs = [d for d in dirs if os.path.isdir(d)]
    files = []
    if len(dirs) == 0:
        #print "No valid directory found!"
        sys.exit(1)
    for d in dirs:
        label = os.path.basename(d.rstrip('/'))

        wavs = glob.glob(d + '/*.wav')
        print (wavs)
        if len(wavs) == 0:
            print ("No wav file found in {0}".format(d))
            continue
        print ("Label {0} has files {1}".format(label,
','.join(wavs)))
        for wav in wavs:
```

```
            fs, signal = read_wav(wav)
            m.enroll(label, fs, signal)

    m.train()
    m.dump(output_model)

def task_predict(input_files, input_model):
    m = ModelInterface.load(input_model)
    for f in glob.glob(os.path.expanduser(input_files)):
        fs, signal = read_wav(f)
        label = m.predict(fs, signal)
        print (f, '->', label)

if __name__ == '__main__':
    global args
    args = get_args()

    task = args.task
    if task == 'enroll':
        task_enroll(args.input, args.model)
    elif task == 'predict':
        task_predict(args.input, args.model)
```

Emotion Recognition:

```
import subprocess
filename="audioAnalysis.py regressionFile -i hahaha.wav --model
svm --regression data/svmSpeechEmotion], stdout=subprocess.PIPE,
shell=True"
proc = subprocess.Popen(["pythonw"+filename])
(out, err) = proc.communicate()
print "program output:", out
arr=out.split()

arr[1]=float(arr[1])
arr[3]=float(arr[3])

if arr[1]>0 and arr[3]>0:
    print "happy"
elif arr[1]>0 and arr[3]<0:
    print "angry"
elif arr[1]<0 and arr[3]>0:
    print "calm"
else:
    print "sad"
```

9.2 Input or Output Listings

Input:

Voice.wav

Training set consists of the audio files of male and females for gender detection. Total dataset consists of around 1000 files.

For emotion recognition we had around 50 audio files for which model is trained based on the features of the voice and it would generate the valence and arousal of the voice file.

Speaker recognition, we took 10 samples of each of our team member and then trained the model to recognize the individual voice

Output:

1. Based on the training set, the output of the gender detection model would be male or female.
2. The output of emotion recognition would be angry,happy,sad and calm based on the generated output of the valence and arousal values.
3. The output of the speaker recognition tells who is the speaker and differentiates between two different speakers.