Dissertation on

"Voice Based Age Detection and Song Suggestion Based On Age Group"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

Submitted by:

Chandhan Bhatta	**01FB16ECS094**
D Ganesh Kumar	**01FB16ECS102**
G Gautham	**01FB16ECS118**

Under the guidance of

Internal Guide
Prof. Kavitha KN
Asst. Professor,
PES University

January – May 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Voice Based Age Detection and Song Suggestion Based On Age Group'

is a bonafide work carried out by

Chandhan Bhatta	**01FB16ECS094**
D Ganesh Kumar	**01FB16ECS102**
G Gautham	**01FB16ECS118**

In partial fulfilment for the completion of eighth semester project work in the Program of Study Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2020 – May. 2020. It is certified that all corrections and suggestions indicated for internal assessment have been successfully incorporated in the report. The dissertation has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature	Signature	Signature
Prof. Kavitha KN	Dr. Shylaja S S	Dr. B K Keshavan
Asst. Professor	Chairperson	Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. ____________________________	____________________________
2. ____________________________	

DECLARATION

I hereby declare that the project entitled "**Voice Based Age Detection and Song Suggestion Based On Age Group**" has been carried out by me under the guidance of Prof. Kavitha KN, Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2020. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

01FB16ECS094 Chandhan Bhatta [illegible]

01FB16ECS102 D Ganesh Kumar [illegible]

01FB16ECS118 G Gautham [illegible]

ACKNOWLEDGEMENT

In completion of my project, I had taken help from our guide and I would like to show my gratitude to Prof.Kavitha KN Dept. of Computer Science, PES University, for her continuous guidance, assistance and encouragement throughout the development of this project.

I am grateful to the project coordinators – Prof. Preet Kanwal , Prof. Sangeetha V I, panel members - Prof.Phalachandra H.L. and Prof. Uma D, for organizing, managing and helping out with the entire process.

I take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department.

I would like to show my gratitude towards Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I would like to show my gratitude towards Dr. Suryaprasad J, Vice Chancellor, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way.

ABSTRACT

Since most of the technologies now are going ahead with voice based models it would be efficient and accurate to detect Age from the voice Sample.

These analysed voice sample can be used in a variety of use cases ranging from demographic analytics to restrict access to certain user based on their age.

We have for this project chosen song recommendation as an use case. The users are recommended songs based on their age classification based on their voice samples.

TABLE OF CONTENTS

CHAPTER-1

INTRODUCTION

With Increase in technology has led to increase in more and more connected devices, which can be remotely operated through voice activation. And in today's day and age results are personalized and targeted towards one's personal choice in order to cater to their actual needs. Thus a voice based age detection system would further help in personalizing one's results without them having to manually having to input their age, which in many cases they may not be willing to do.

Voice Based Age Detection System can be implemented in many cases as enlisted below:-

- Age detection can be used in devices which use voice controlled tools like Apple's Siri, Amazon's Alexa etc.
- Music enthusiast
- Music scientist, researchers
- Call centres can analyse their demography better.

Also detection of age based on voice can also help certain age groups from accessing content or data that is restricted for their use. It is a more fool proof system than ones currently in use.

CHAPTER 2

PROBLEM DEFINITION

With remote voice control of smart devices such as Amazon Alexa or Google Home, user can play music just at their voice command. Thus to suggest music which are more in the age group of the given user an age analysis could help the smart assistant serve better results.

Users can be grouped into age groups of 10-20(teens), 20-30(twenties),30-40(thirties),40-50(forties),50-60(fifties),60-70(sixties),70-80(seventies),80-90(eighties),90-100(nineties).

The classified age group of the User is then use to Recommend music based on the popularity in that age group and also prevent underage users from listening to explicit content.

The User voice recording can be passed to the system which classifies the user into respective age group and suggests songs that are more popular in that age group.

CHAPTER-3

LITERATURE SURVEY

3.1 **Convolutional And recurrent Neural Network used for Age and Gender Prediction from Speech**

This Paper Discusses in depth the usage of neural networks in order to predict age and gender of the user based on their voice sample.

The paper uses the same Mozilla Commons dataset as used by our team. But the users are classified into Young, Aged and Elderly.

They could get an accuracy of **79.51%.**

3.1.1 Pre Processing

The first sound records show up in mp3 format, with a piece each of size 64kbps. The first step is to change over them to pulse code modulation, with 8kHz examining rate, to estimate the nature of phone lines. It must be understood that the annoyance due to mp3 format and impacts majorly to high recurrence parts of the sound signals, and it is normal that a large chunk of the data in the 4kHz band of phone signals stays accessible. The sounds recorded contain emptiness in the start and the end, that must be erased to evade the confusion. A Voice checking detector has been executed, in view of vitality in the sound sign. All the sound signs have been standardized, with the goal that amplitudes should be in the range plus one and minus one. At this stage, the Short and Time Fourier Transform has been applied to the audio signals, with samples of 20ms (160 examples of the length at 8kHz examining recurrence), covering the half of their lengths (100 edges for each time in seconds). The last step consists of finding the energy of sub groups in the Mel-scale.

3.1.2. Age and Gender Detection Based on Deep Learning

The detection depends on a convolutional and intermittent neural system, which has been created using keras, the Python Inbuilt Deep Learning Library. keras is an tool used to create neural systems using API, written in Python and built for running on tensorflow, cntk, or theano. It was made to make work really quick. The detection mechanism which has been made using keras functionalities has been acquired after a thorough hunt of the best design. At next stage, the best results are acquired with the mechanism that is portrayed below, which is made out of two two dimensional convolutional layers, one repetitive layer, and a yield layer. Its structure is portrayed in Fig. 2, which is clarified as follows:

1. The first convolutional layer convolves the info network with a spatial channel of size [3x3]. An all out number of 48 channels are thought of. In the event that the information is a [100x20] grid, the yield is a 3D lattice with measurements [100x20x48].

2. The subsequent advance executes max pooling with [1x2] channels and step 2, offering ascend to a lattice of measurements [100x10x48].

3. The yield of the MaxPool layer is applied to another 2D convolutional later, with 24 [3x3] channels.

4. Max pooling with [1x2] channels and step 2, is applied once more.

5. The yield of the Max pooling layer is reshaped, to offer ascent to bi-dimensional networks, of size [100x120], keeping up a similar data.

6. The reshaped network is then applied to a Gated Recurrent Unit (GRU), which was proposed by Cho et al. [14] to make each intermittent unit to adaptively catch conditions of various time scales. GRUs are improved form of standard repetitive neural system. To tackle the evaporating inclination issue of a standard RNN, GRUs use, alleged, update entryway and reset door, two vectors which choose what data ought to be passed to the yield, and that can be prepared to keep data from some time in the past, without washing it through time or evacuate data which is superfluous to the expectation .

7. The yields of the GRU layer are consolidated to create to conclusive yields, utilizing capacity Dense. Thick actualizes the activity: yield = activation(dot(input, portion) + inclination), where actuation is the component astute enactment work went as the initiation contention, piece is a loads framework made by the layer, and predisposition is a predisposition vector made by the layer.

The yield layer has three yields for age bunch estimation, and another for sexual orientation differentiation. The contribution to the estimator is determined each second, and it will be a framework with 100 lines (number of 20ms length casings), and 20 segments .

3.2 Using Voice to detect Age and Gender Using the Learning Generative Sparse Models

This Paper Discusses in depth the usage of SVM and GMM in order to predict age of the user based on their voice sample.

This paper uses the same Mozilla Commons dataset as used by our team. Also used 200 volunteers to test the data.

They could get an accuracy of **73.89%.**

3.2.1 Pre Processing

The input signal should be transferred into a feature domain to simplify the signal analysis with maintaining the features. The MFCC feature domain is a good selection to present the most important content of the voice signals in a compact form.

3.2.2 Implementation Details

The proposed recognition method is assessed using different evaluation measures using SpeechData II corpus that is a telephone speech databases with gender and age labels [20]. This large multi-talker data base has been used for research about speech recognition and speaker verification problem that includes words or sentences read by 5000 speakers of both genders. The training and test sets include 100 speakers, 70 and 30 speakers in the training and testing aspects. The sampling rate of the input data is 8 kHz and feature vectors are yielded using 39 MFCC coefficients. The number of Mel filters is set to 12. The signals are broken down with 37.5 ms frame length and 40% intersection with a Hamming window. All framing, pre-processing steps are the same in the training aspect and test aspect of different speakers. The defined parameters in different steps of Algorithm 1 are assigned according to the experimental evaluations. The value of ε parameters is set to 0.01 and 0.015 for male and female speakers. Also, the value of $\varepsilon 1$ is adjusted to 0.01 for both male and female speakers. N, $\mu 0$, K are set to 10, 0.2 and 8 for all training sets, respectively.

3.3 Using Deep Neural Network for Age Detection

This Paper Discusses in depth the usage of i-vectors in order to predict age of the user based on their voice sample.

This Project uses the NIST SRE-08 dataset.

They could get an accuracy of 84% using i-vectors and 99% using x-vectors.

3.3.1 x-Vector for neural network

The x-Vector neural system model is depicted in detail in this segment. The highlights were 23-diminish MFCC brief timeframe mean standardized over sliding the window of 3 seconds. A vitality based SAD (discourse action identification) was utilized to expel non-discourse outlines. The DNN design is illustrated in table 1. The time-postpone profound system layer (TDNN) with the amended direct unit (ReLU) non-linearity was utilized. Group standardization was likewise utilized after the non-linearity. The initial 4 layers had little fleeting setting focused at the present casing t, for example layer with [−3, 0, +3] connects outlines from t−3, t to t+3 and manufactures the layer on them. The insights pooling layer totals data across time over all T outline level yields at layer 4. The registered mean and standard deviation are connected and proliferated through next layers and the yield softmax layer. The preparation models comprised of smaller than normal bunches of discourse ut-terances with the comparing age mark. To make the system strong to variable-length test expressions, we prepared the DNN model utilizing variable length pieces haphazardly inspected from the full-length articulations. The impact of fixed versus variable length preparing lump length is examined in subtleties in Section 5.2. Since x-vector is information eager methodology, we additionally utilized information aug-mentation, comprising in adding commotion and resonation to in-wrinkle the measure of preparing information, which improved the mean outright mistake from 5.9 to 4.9.

3.3.2 i-Vector for neural network

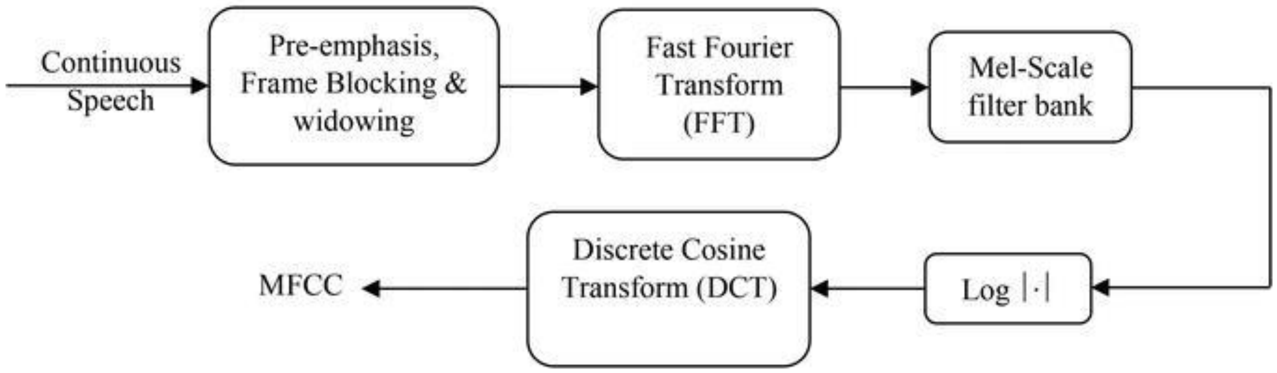
The i-vector neural system model likewise utilized MFCC highlights with brief timeframe focusing. The UBM and i-vector extractors were prepared on NIST SRE04-06 English phone discourse containing 1936 female speakers and 679 male speakers. We utilized a 2048 com-ponent GMM-UBM model with full covariance grids. Absolute inconstancy subspace measurement was set to 400. It merits referencing that there is no speaker cover between the information used to prepare the I-vector extractor and information used to prepare and test the age estimation backend/x-Vector framework. The info measurement for the DNN back-end was 400 for i-vectors and 800 for the combination tests link I-vector and x-vector. We likewise tested applying LDA to the info embeddings, the LDA measurement was set to 50.

The LDA change was found out regarding the age names. We split the SRE08 information into train and approval sets with-out speaker cover to preparing the back-end. All the speakers with in excess of 6 expressions were utilized to prepare the back-end. The various speakers were utilized for approval. To be consis-tent with the x-vector explores, all ages with hardly any preparation tests were mapped to the past age class. Each concealed layer had 256 neurons with sigmoid non-linearity. The yield layer had one straight neuron to anticipate the age esteem. The system was prepared to limit mean-squared-mistake objective. The little cluster size was set to 32. We utilized stochastic slope plunge (SGD) streamlining agent with an underlying learning pace of 0.001 and a force of 0.9. We diminished the learning rate by a factor of 2 when the approval misfortune doesn't improve for two progressive ages. Least learning rate was set to 1e-05. Preparing is halted if approval doesn't im-demonstrate for three successive ages. The model with best vali-dation misfortune was utilized for testing.

3.4 A Review on Feature Extraction and Noise Reduction Technique

This paper provides outline various feature extraction and noise reduction technique. This paper helps in selecting the technique beside on their relative advantages & disadvantages. This paper concludes with the choice on feature direction for developing technique in human pc interface system.

MFCC: Mel-Frequency Cepstral Coefficients


Continuous Speech
Pre-emphasis, Frame Blocking & widowing
Fast Fourier Transform (FFT)
Mel-Scale filter bank
Log |·|
Discrete Cosine Transform (DCT)
MFCC

MFCC values are not very strong in the presence of too much noise, and so we take those values and get the normalized version of it , use this normalized values in speech recognition systems to lessen the influence of noise.

CHAPTER 4

PROJECT REQUIREMENT SPECIFICATION

4.1 Introduction

This document relates to the background and surrounding information regarding our project – Voice Based Age Detection and Song Recommendation. It deals with the scope, shortcomings, risks, architecture, etc. of the project and is meant to serve as documentation to the end user who wishes to understand the project in detail and modify it to achieve better results.

4.2 Purpose

The purpose of this project is to build a web interface wherein we'll be able to record voice of a user and classify them into a age group, based which songs will be recommended to user, and those songs can be played using the UI.

4.3 Modules

4.3.1. Pre-processing the dataset

The Dataset used for age detection is Mozilla's open source dataset known as Common Voice which contains 36GB of labelled data.
Pre-processing of data starts by converting the audio files to .wav format using FFMPG feature.
Then we use pyaudioanalysis tool to extract 34 features various statistics such as mean, median, mode etc. are extracted totalling to around 170(34*5) features being extracted.

4.3.2. Additional utilities

We have constructed a simple baseline line model to to test for the best feature extraction model among 3 major models namely, PCA based Feature Extraction, Tree Based Feature Extraction, and Chi Square Based Feature Extraction.
Since Tree Based Model gave higher accuracy over the baseline model we proceeded with tree based model for the final model.

Another additional utility is a song recommendation system which recommends the most popular song among a particular age group to the user.

4.3.3. The Model

Baseline Model to test for feature Extraction:- The model to choose correct feature selector , 2 layer neural network, with a hidden layer size of 20. There is a dropout layer with a 0.75 just after the hidden layer and before the output softmax layer. We use a standard scaler because our features are on different scales which will ensure we get more efficient/effective error minimization.
Final Model for Age detection :-Our basic architecture for the model was an input layer with nodes equal to the number of features (~80) and an output layer equal to the number of classes (8). There are four hidden layers. The first one has 128 nodes, the second has 256, the third has 256, and the fourth has 128. Because there are is a relatively high number of layers, as well as nodes, the neural network can make more complex and accurate predictions.

4.3.4. Train and Test

Since the dataset was large we had to break down the dataset into smaller batches to trains the data more efficiently.
We used powerful python tools such as keras, tensorflow etc. to build the model.

4.4 Intended Audience

The project in it’s current scope can be used by voice operated music player systems in order to recommend songs to users over inter connected IOT devices. The project provides a simple UI for the user to easily record their voice and the songs are then recommended based on the age group of the user.

The Age Detection part of the project can also be used in call centres to do an analysis of their demographic in order for them to understand the demographic better and cater to their needs. It can also be extended as fool proof system to prevent users from certain age groups from accessing certain data or content.

4.5 Project Scope

1) The project Detects age of the user based on the voice.
2) The age is then used to recommend songs based on the popularity within an age group.
3) Provides a simple ui to record voice and a simple song player. ui.

4.6 Product Perspective

1. This project is independent and self-contained. It will consist of a web interface to summarise a given block of text which will be done by the backend component deployed on our local machine.
2. Software platforms used for development -Google Colab, Tensorflow, Keras,Pyauidoanalysis,MFCC,FFMPG,Scikit learn and pandas.
3. Software platforms used for deployment - Xampp server and Flask will be needed to deploy this project on the local system.

4.7 Special Characteristics

1) Age detection based on voice.
2) Choosing most apt feature selector.

4.8 Help

A user manual shall be provided to the end users to ensure adequate instructions are provided on the normal operation of the application and also in certain cases of unplanned breakdown.

4.9 General Constraints, Assumptions and Dependencies

Constraints :

- Voice recording should be more than 3 seconds long.
- The system will probably be limited to one user since it will be hosted locally.
- There must not be any noise in recording.
- Doesn’t work for children below age 13.
- Assumes user music taste is inline with popular music among their age group.

Assumptions :

- Voice recording is more than 3 seconds.
- There will not be any noise in recording.
- The user is above age of 13.
- Users choice is inline with their age group.

4.10 Safety Requirements

- Children below age of 13 shouldn’t be allowed to use as the system may wrongly classify them and play inappropriate songs.
- No Physical safety standards.

4.11 Risks

- The age predicted might not be accurate.
- The songs recommended might not be inline with users choice.
- Misuse of system by playing others voice is possible.

CHAPTER 5

SYSTEM REQUIREMENT SPECIFICATION

5.1 Functional Requirements :

5.1.1 Age Recommendation Requirements :

- The system must be callable of recording voices.
- The Recorded voice must then be converted to .wav format.
- Necessary features must then be extracted from the audio clipping.
- To select the best Feature selector a baseline model must be created to choose the best Feature extractor.
- A more robust model is then built in comparison to the baseline model.
- The features are extracted using the most optimum feature extractor and passed to the model for training.
- The output of the predicted age is passed to song recommendation system.

5.1.2 Web based/UI Requirements :

- A login Page
- Ui to register new user.
- A Simple Ui to Record voice of user.
- Ui to view and play the recommended songs.

5.1.3 Song Recommender Requirements :

- The Song recommender takes age group as an input
- Recommends the most popular songs within that age group.

5.2 Non-Functional Requirements :

5.2.1 Usability

The UI is a simple and ready to use soulution.The user can just login or register if they are a new user. Upon login the audio of the user is recorded. Further the audio sample is analysed to get a age group based on which songs are recommeneded. A simple UI is provided to play the songs recommended to the user.

5.2.2 Reliability

Since the project uses highly powerful neural netwroks and machine learning algorithms. Though the accuracy of age prediction is around 85 percent, there is a chance there might be a wrong prediction especially in age groups above 40.

The song recommendation is a straight forward popularity based recommender.

5.2.3 Performance

The training of model and feature extraction takes take 12 hrs+ time due to the large dataset(~36 GB). Once the data is trained it is easy to

5.2.4 Reusability/Interoperability

This may be a future venture but we would like the hidden underlying model to basically be extendible to other datasets(accompanied with glove vectors) irrespective of their size.

5.3 Other Requirements :

Since Machine learning algorithms take large periods of time to train the data, usage of software like CUDA which uses the graphics card to run the machine learning can speed up the process. Also use of open source tools like Google Colab can be used to speed up the machine learning process.

To further speed up the process the computing can be performed on Amazon AWS's instances.

5.3.1 Hardware Requirements :

The requirements need to be:

1) A Device With High Computing Power.
2) 8GB RAM
3) 50 GB+ ROM
4) Dedicated Graphics Card
5) AWS ML Instances(Optional)

5.3.2 Software Requirements :

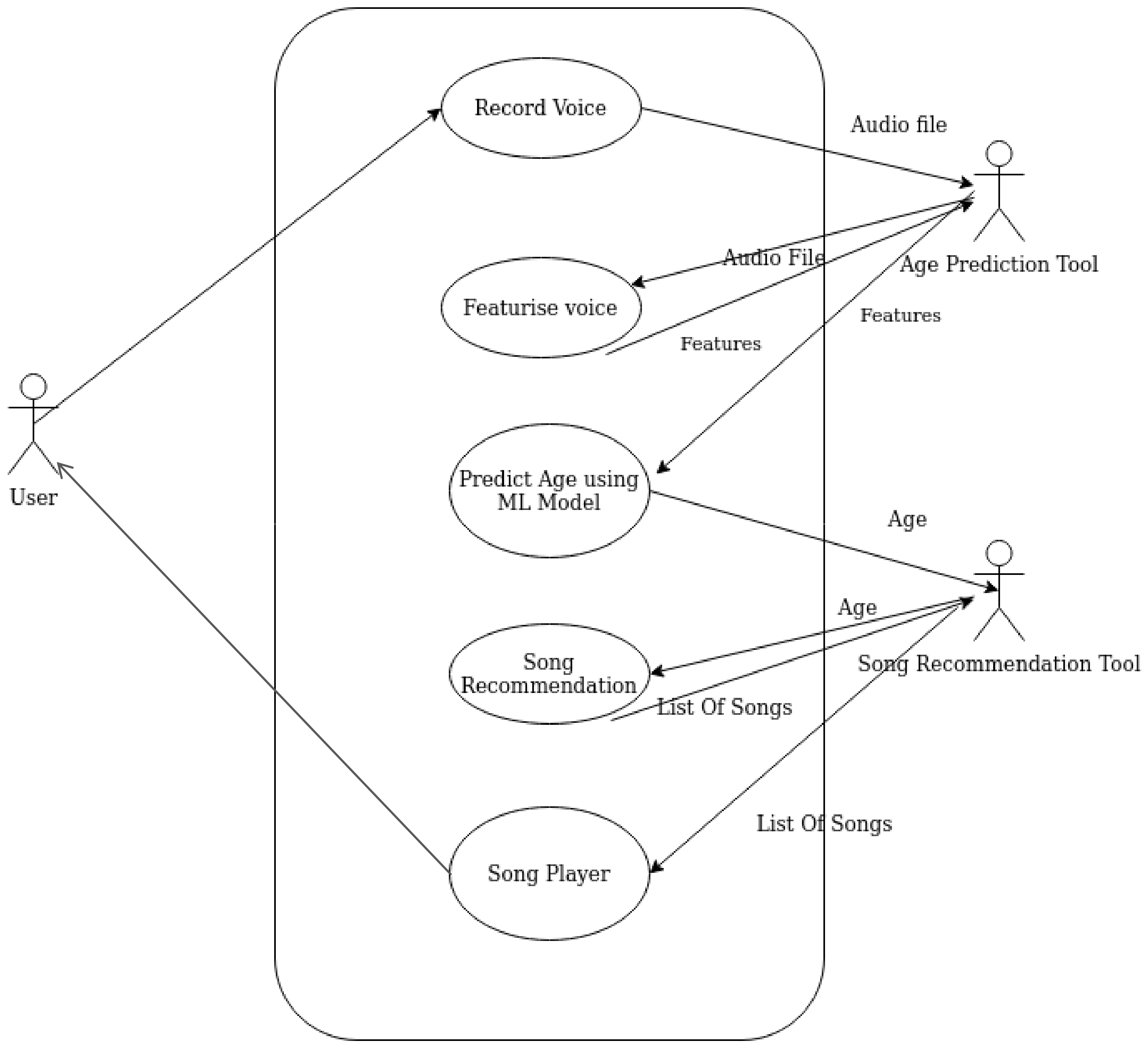
The software which would be expected to be pre installed would include :

1) OS
2) Flask - Rest API
3) Python & primarily the following libraries : nltk,numpy,sklearn.
4) TensorFlow/Keras for building the model
5) Google Colab
6) Jupyter Notebook
7) Cuda
8) HTML, CSS, Javascript

5.3.3 Communication Interfaces :

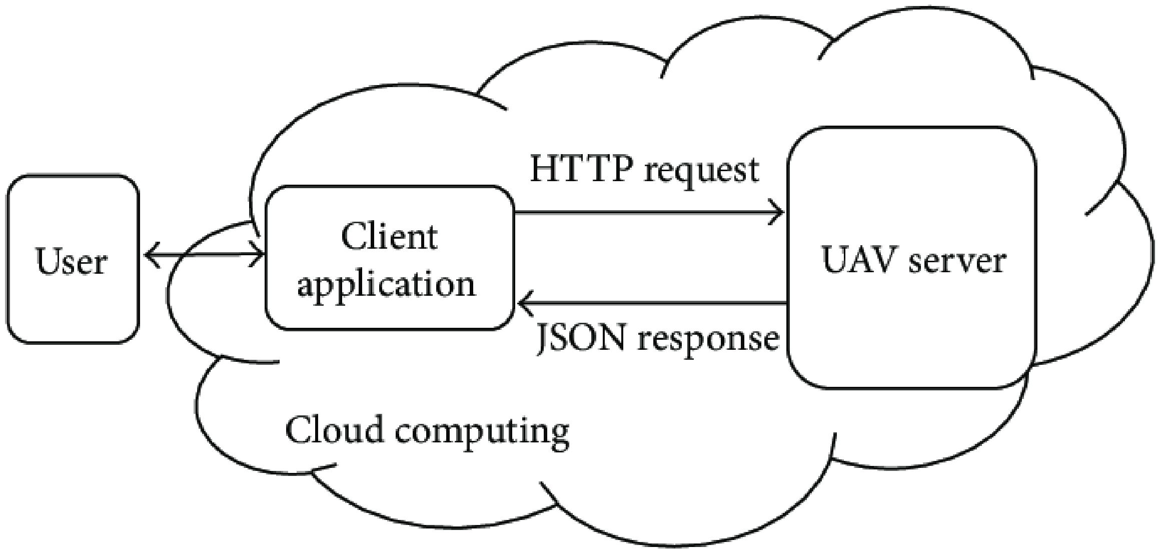
Communication primarily exists between UI and server(flask) which holds the model. The UI sends a request to FLASK server that uses the model and utils to predict the summary. The underlying protocol is going to be HTTP. With GET and POST requests basically being sent over.
First The File Path of the audio recording is sent to age predictor via flask POST operation. The age prediction is then passed to song recommender. Once The recommendations are ready they are fetch and displayed using flask GET operation.

5.4 Use case Diagram :



CHAPTER 6
SYSTEM DESIGN

Our System follows the client server architecture. The audio is fed into the interface located on the client and the request is processed by the server. The songs are then recommended to user. This is a 3 layer architecture since we also have files that contain other articles which serve as the database.



The code on the server which consists of the following components:

- Pre-processing.
- Model
- Utils
- UI
- Server

These are organized according to the modular design approach as a structured approach enables us to analyse the inner workings and pin down errors quicker. Furthermore, each model can be executed as a unit and can result in better performance. Example: the testing module is executed separately.

Web application is provided where has to login to use it, where user is prompted to record voice , this audio file is used for age prediction.

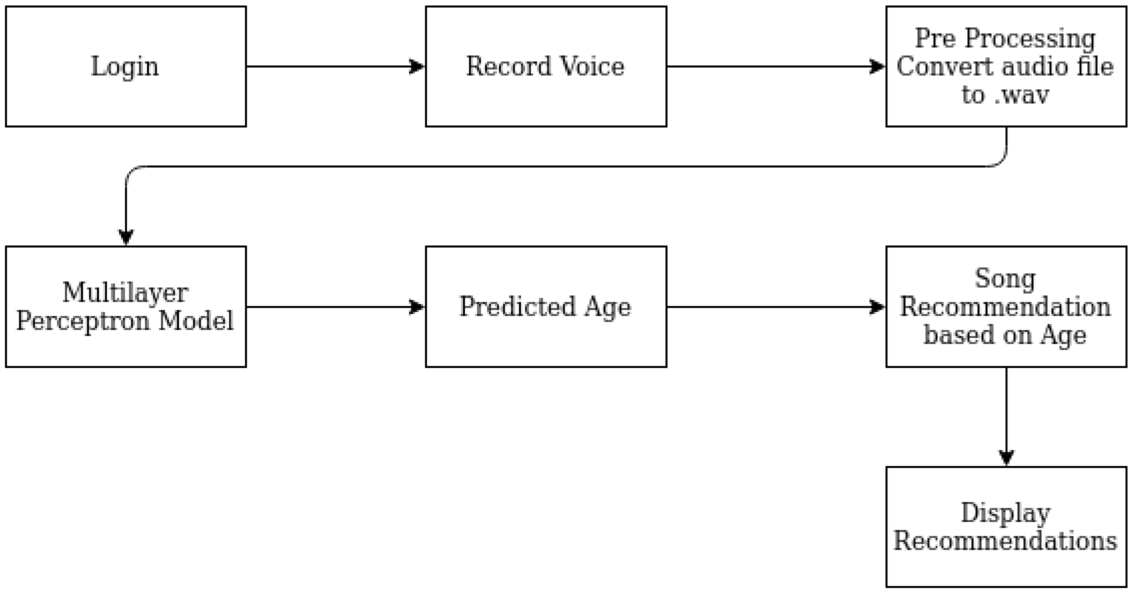
Age is predicted into using multilayered perceptron model which consist of pyaudioanalysis tool which helps featurise the audio file.

Song is suggested based on the age group that user is predicted.

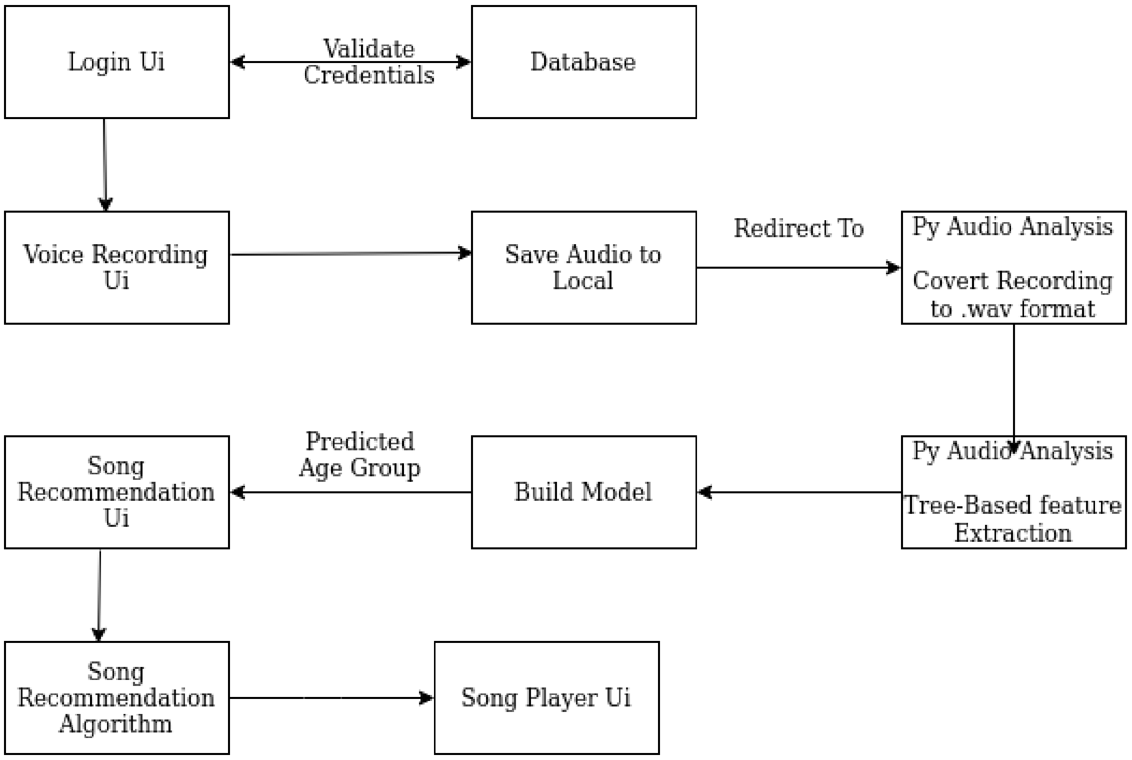
The list of song suggested are displayed to the user in web app.

CHAPTER-7
DETAILED SYSTEM DESIGN

7.1 Class Diagram


Login
Record Voice
Pre Processing Convert audio file to .wav
Multilayer Perceptron Model
Predicted Age
Song Recommendation based on Age
Display Recommendations

7.2 Architecture



CHAPTER-8
Implementation And Pseudo Code

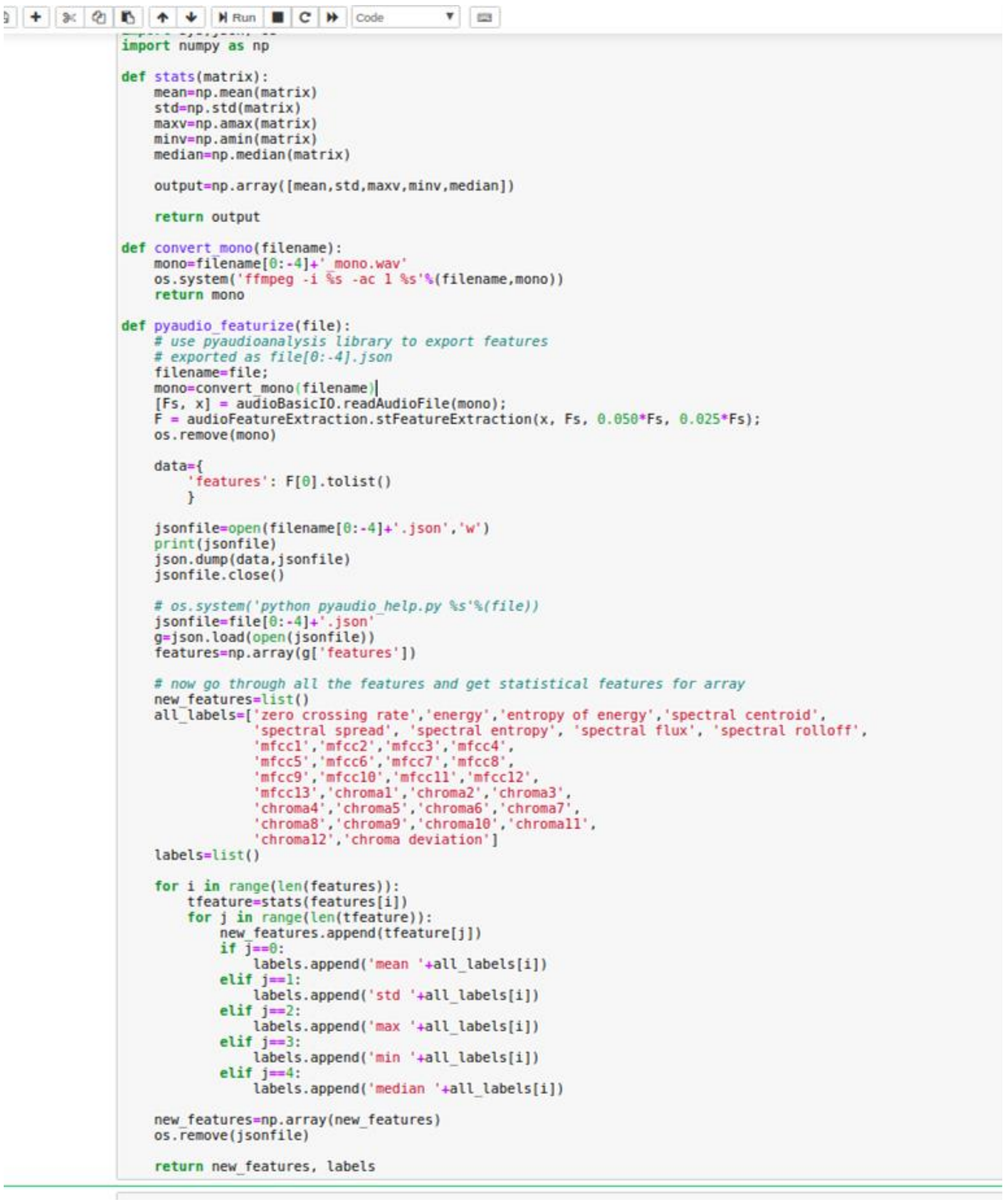
Web application where has to login to use it, where user is prompted to record voice , this audio file is used for age prediction.

Age is predicted into using multi-layered perceptron model which consist of pyaudioanalysis tool which helps featurise the audio file.

Song is suggested based on the age group that user is predicted.

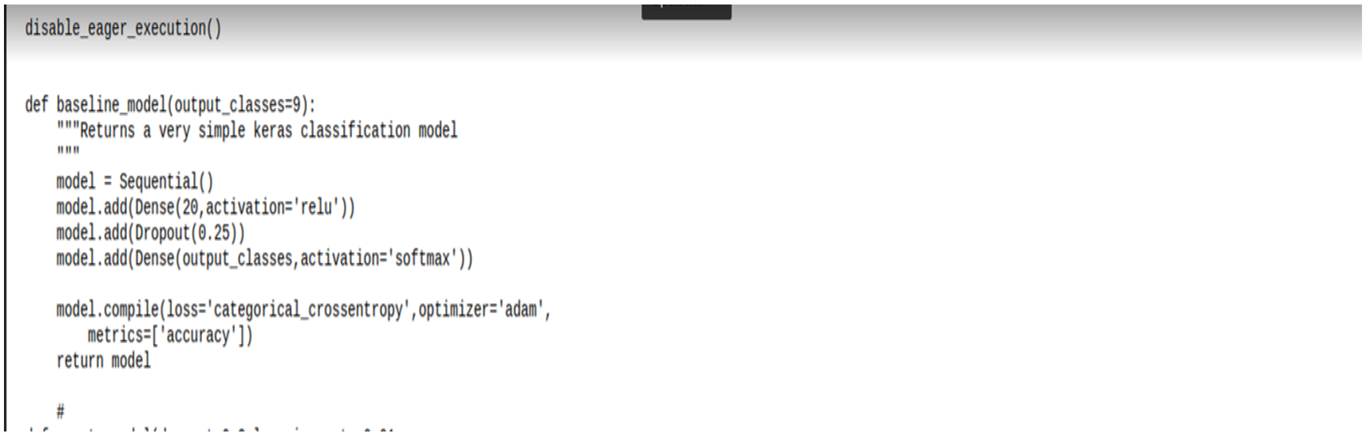
The list of song suggested are displayed to the user in web app.
reprocessing for Age detection includes converting audio file to wav , featurizing the audio file using python audio analysis tool , this tool gets common audio features like mel spectrogram feature coefficients(MFCC’s), spectral centroid, spectral contrast, and spectral rolloff etc .
we compute them in intervals and then take the mean,median, maximum, minimum, and standard deviation of all the extracted values, this includes 35 features , 5 statistical components of each 170 total features.
Later we pass these features to multi-layered perceptron model includes an input layer with nodes equal to the number of features (~80) and an output layer equal to the number of classes (9). There are four hidden-layers.
Using tensorflow , keras modules we built the multi-layered perceptron model, using is MP we predict the which Age group class the user belongs to .
For song suggestion we have built our own recommender model that takes song ID and user’s age group and suggest song based on it .

Pre Processing Pseudo Code



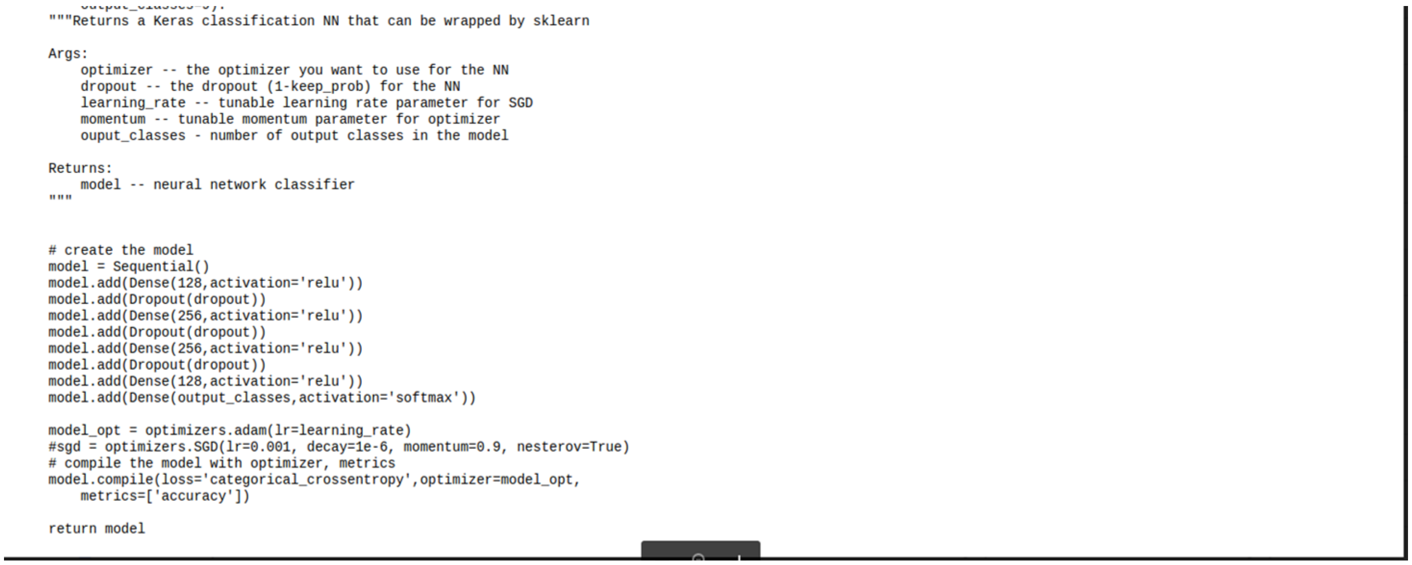
```
import numpy as np

def stats(matrix):
    mean=np.mean(matrix)
    std=np.std(matrix)
    maxv=np.amax(matrix)
    minv=np.amin(matrix)
    median=np.median(matrix)

    output=np.array([mean,std,maxv,minv,median])

    return output

def convert_mono(filename):
    mono=filename[0:-4]+'_mono.wav'
    os.system('ffmpeg -i %s -ac 1 %s'%(filename,mono))
    return mono

def pyaudio_featurize(file):
    # use pyaudioanalysis library to export features
    # exported as file[0:-4].json
    filename=file;
    mono=convert_mono(filename)
    [Fs, x] = audioBasicIO.readAudioFile(mono);
    F = audioFeatureExtraction.stFeatureExtraction(x, Fs, 0.050*Fs, 0.025*Fs);
    os.remove(mono)

    data={
        'features': F[0].tolist()
        }

    jsonfile=open(filename[0:-4]+'.json','w')
    print(jsonfile)
    json.dump(data,jsonfile)
    jsonfile.close()

    # os.system('python pyaudio_help.py %s'%(file))
    jsonfile=file[0:-4]+'.json'
    g=json.load(open(jsonfile))
    features=np.array(g['features'])

    # now go through all the features and get statistical features for array
    new_features=list()
    all_labels=['zero crossing rate','energy','entropy of energy','spectral centroid',
                'spectral spread', 'spectral entropy', 'spectral flux', 'spectral rolloff',
                'mfcc1','mfcc2','mfcc3','mfcc4',
                'mfcc5','mfcc6','mfcc7','mfcc8',
                'mfcc9','mfcc10','mfcc11','mfcc12',
                'mfcc13','chroma1','chroma2','chroma3',
                'chroma4','chroma5','chroma6','chroma7',
                'chroma8','chroma9','chroma10','chroma11',
                'chroma12','chroma deviation']
    labels=list()

    for i in range(len(features)):
        tfeature=stats(features[i])
        for j in range(len(tfeature)):
            new_features.append(tfeature[j])
            if j==0:
                labels.append('mean '+all_labels[i])
            elif j==1:
                labels.append('std '+all_labels[i])
            elif j==2:
                labels.append('max '+all_labels[i])
            elif j==3:
                labels.append('min '+all_labels[i])
            elif j==4:
                labels.append('median '+all_labels[i])

    new_features=np.array(new_features)
    os.remove(jsonfile)

    return new_features, labels
```

BaseLine Model For Choosing Feature Extractor



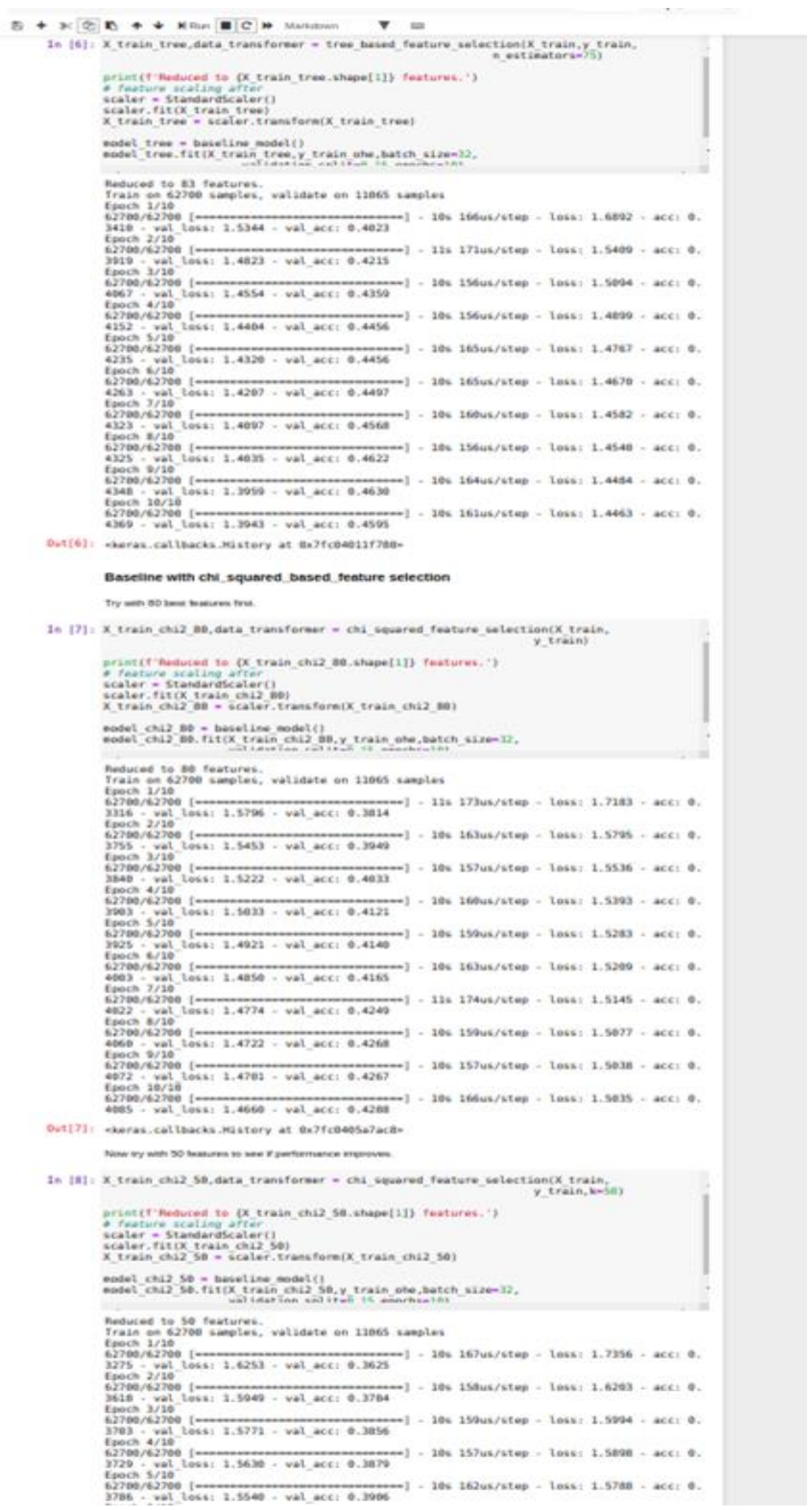
```
disable_eager_execution()


def baseline_model(output_classes=9):
    """Returns a very simple keras classification model
    """
    model = Sequential()
    model.add(Dense(20,activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(output_classes,activation='softmax'))

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

    #
```

Final Model



```
"""Returns a Keras classification NN that can be wrapped by sklearn

Args:
    optimizer -- the optimizer you want to use for the NN
    dropout -- the dropout (1-keep_prob) for the NN
    learning_rate -- tunable learning rate parameter for SGD
    momentum -- tunable momentum parameter for optimizer
    ouput_classes - number of output classes in the model

Returns:
    model -- neural network classifier
"""


# create the model
model = Sequential()
model.add(Dense(128,activation='relu'))
model.add(Dropout(dropout))
model.add(Dense(256,activation='relu'))
model.add(Dropout(dropout))
model.add(Dense(256,activation='relu'))
model.add(Dropout(dropout))
model.add(Dense(128,activation='relu'))
model.add(Dense(output_classes,activation='softmax'))

model_opt = optimizers.adam(lr=learning_rate)
#sgd = optimizers.SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True)
# compile the model with optimizer, metrics
model.compile(loss='categorical_crossentropy',optimizer=model_opt,
    metrics=['accuracy'])

return model
```

Feature Extraction



```
In [6]: X_train_tree,data_transformer = tree_based_feature_selection(X_train,y_train,
                                                                n_estimators=75)

        print(f'Reduced to {X_train_tree.shape[1]} features.')
        # feature scaling after
        scaler = StandardScaler()
        scaler.fit(X_train_tree)
        X_train_tree = scaler.transform(X_train_tree)

        model_tree = baseline_model()
        model_tree.fit(X_train_tree,y_train_ohe,batch_size=32,
                      validation_split=0.15,epochs=10)

        Reduced to 83 features.
        Train on 62700 samples, validate on 11065 samples
        Epoch 1/10
        62700/62700 [==============================] - 10s 166us/step - loss: 1.6892 - acc: 0.
        3410 - val_loss: 1.5344 - val_acc: 0.4023
        Epoch 2/10
        62700/62700 [==============================] - 11s 171us/step - loss: 1.5409 - acc: 0.
        3919 - val_loss: 1.4823 - val_acc: 0.4215
        Epoch 3/10
        62700/62700 [==============================] - 10s 156us/step - loss: 1.5094 - acc: 0.
        4067 - val_loss: 1.4554 - val_acc: 0.4350
        Epoch 4/10
        62700/62700 [==============================] - 10s 156us/step - loss: 1.4899 - acc: 0.
        4152 - val_loss: 1.4404 - val_acc: 0.4456
        Epoch 5/10
        62700/62700 [==============================] - 10s 165us/step - loss: 1.4767 - acc: 0.
        4235 - val_loss: 1.4320 - val_acc: 0.4456
        Epoch 6/10
        62700/62700 [==============================] - 10s 165us/step - loss: 1.4670 - acc: 0.
        4263 - val_loss: 1.4207 - val_acc: 0.4497
        Epoch 7/10
        62700/62700 [==============================] - 10s 160us/step - loss: 1.4582 - acc: 0.
        4323 - val_loss: 1.4097 - val_acc: 0.4568
        Epoch 8/10
        62700/62700 [==============================] - 10s 156us/step - loss: 1.4540 - acc: 0.
        4325 - val_loss: 1.4035 - val_acc: 0.4622
        Epoch 9/10
        62700/62700 [==============================] - 10s 164us/step - loss: 1.4484 - acc: 0.
        4348 - val_loss: 1.3959 - val_acc: 0.4630
        Epoch 10/10
        62700/62700 [==============================] - 10s 161us/step - loss: 1.4463 - acc: 0.
        4369 - val_loss: 1.3943 - val_acc: 0.4595

Out[6]: <keras.callbacks.History at 0x7fc04011f780>
```

Baseline with chi_squared_based_feature selection

Try with 80 best features first.

```
In [7]: X_train_chi2_80,data_transformer = chi_squared_feature_selection(X_train,
                                                                      y_train)

        print(f'Reduced to {X_train_chi2_80.shape[1]} features.')
        # feature scaling after
        scaler = StandardScaler()
        scaler.fit(X_train_chi2_80)
        X_train_chi2_80 = scaler.transform(X_train_chi2_80)

        model_chi2_80 = baseline_model()
        model_chi2_80.fit(X_train_chi2_80,y_train_ohe,batch_size=32,
                         validation_split=0.15,epochs=10)

        Reduced to 80 features.
        Train on 62700 samples, validate on 11065 samples
        Epoch 1/10
        62700/62700 [==============================] - 11s 173us/step - loss: 1.7183 - acc: 0.
        3316 - val_loss: 1.5796 - val_acc: 0.3814
        Epoch 2/10
        62700/62700 [==============================] - 10s 163us/step - loss: 1.5795 - acc: 0.
        3755 - val_loss: 1.5453 - val_acc: 0.3949
        Epoch 3/10
        62700/62700 [==============================] - 10s 157us/step - loss: 1.5536 - acc: 0.
        3840 - val_loss: 1.5222 - val_acc: 0.4033
        Epoch 4/10
        62700/62700 [==============================] - 10s 160us/step - loss: 1.5393 - acc: 0.
        3903 - val_loss: 1.5033 - val_acc: 0.4121
        Epoch 5/10
        62700/62700 [==============================] - 10s 159us/step - loss: 1.5283 - acc: 0.
        3925 - val_loss: 1.4921 - val_acc: 0.4140
        Epoch 6/10
        62700/62700 [==============================] - 10s 163us/step - loss: 1.5209 - acc: 0.
        4003 - val_loss: 1.4850 - val_acc: 0.4165
        Epoch 7/10
        62700/62700 [==============================] - 11s 174us/step - loss: 1.5145 - acc: 0.
        4022 - val_loss: 1.4774 - val_acc: 0.4249
        Epoch 8/10
        62700/62700 [==============================] - 10s 159us/step - loss: 1.5077 - acc: 0.
        4060 - val_loss: 1.4722 - val_acc: 0.4268
        Epoch 9/10
        62700/62700 [==============================] - 10s 157us/step - loss: 1.5038 - acc: 0.
        4072 - val_loss: 1.4701 - val_acc: 0.4267
        Epoch 10/10
        62700/62700 [==============================] - 10s 166us/step - loss: 1.5035 - acc: 0.
        4085 - val_loss: 1.4660 - val_acc: 0.4288

Out[7]: <keras.callbacks.History at 0x7fc0405a7ac8>
```

Now try with 50 features to see if performance improves.

```
In [8]: X_train_chi2_50,data_transformer = chi_squared_feature_selection(X_train,
                                                                      y_train,k=50)

        print(f'Reduced to {X_train_chi2_50.shape[1]} features.')
        # feature scaling after
        scaler = StandardScaler()
        scaler.fit(X_train_chi2_50)
        X_train_chi2_50 = scaler.transform(X_train_chi2_50)

        model_chi2_50 = baseline_model()
        model_chi2_50.fit(X_train_chi2_50,y_train_ohe,batch_size=32,
                         validation_split=0.15,epochs=10)

        Reduced to 50 features.
        Train on 62700 samples, validate on 11065 samples
        Epoch 1/10
        62700/62700 [==============================] - 10s 167us/step - loss: 1.7356 - acc: 0.
        3275 - val_loss: 1.6253 - val_acc: 0.3625
        Epoch 2/10
        62700/62700 [==============================] - 10s 158us/step - loss: 1.6203 - acc: 0.
        3618 - val_loss: 1.5949 - val_acc: 0.3784
        Epoch 3/10
        62700/62700 [==============================] - 10s 159us/step - loss: 1.5994 - acc: 0.
        3703 - val_loss: 1.5771 - val_acc: 0.3856
        Epoch 4/10
        62700/62700 [==============================] - 10s 157us/step - loss: 1.5898 - acc: 0.
        3729 - val_loss: 1.5630 - val_acc: 0.3879
        Epoch 5/10
        62700/62700 [==============================] - 10s 162us/step - loss: 1.5788 - acc: 0.
        3786 - val_loss: 1.5540 - val_acc: 0.3906
```

CHAPTER-9
TESTING

The results and output of the system were manually tested to ensure output and to produce more accurate results

9.1 Test tool

The script has a few functionalities -

- Loading the model
- Passing a recording to model.
- Verify if age if predicted right.
- Verify if most popular songs for that age group are recommended.

9.2 Strategy

The strategy followed for this project is adhoc testing wherein we feed different inputs to the model and verified the correctness of the output.

9.3 Environment

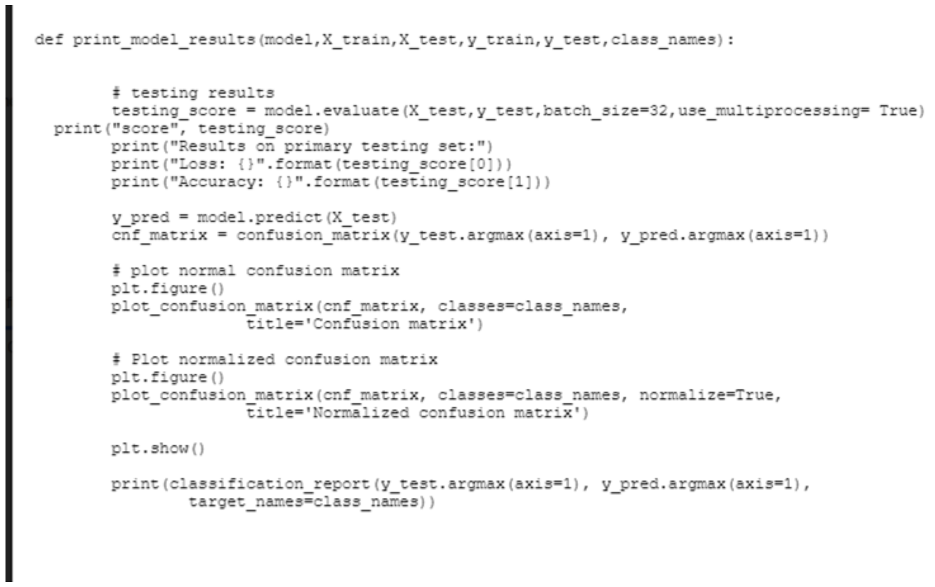
Hardware Environment:-

1) A Device With High Computing Power.
2) 8GB RAM
3) 50 GB+ ROM
4) Dedicated Graphics Card
5) AWS ML Instances(Optional)

Hardware Environment :-

1) Flask - Rest API
2) Python & primarily the following libraries : nltk,numpy,sklearn.
3) TensorFlow/Keras for building the model
4) Google Colab
5) Jupyter Notebook
6) Cuda
7) HTML, CSS, Javascript

Testing Code Snippet



```
def print_model_results(model,X_train,X_test,y_train,y_test,class_names):


        # testing results
        testing_score = model.evaluate(X_test,y_test,batch_size=32,use_multiprocessing= True)
  print("score", testing_score)
        print("Results on primary testing set:")
        print("Loss: {}".format(testing_score[0]))
        print("Accuracy: {}".format(testing_score[1]))

        y_pred = model.predict(X_test)
        cnf_matrix = confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1))

        # plot normal confusion matrix
        plt.figure()
        plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title='Confusion matrix')

        # Plot normalized confusion matrix
        plt.figure()
        plot_confusion_matrix(cnf_matrix, classes=class_names, normalize=True,
                      title='Normalized confusion matrix')

        plt.show()

        print(classification_report(y_test.argmax(axis=1), y_pred.argmax(axis=1),
                target_names=class_names))
```

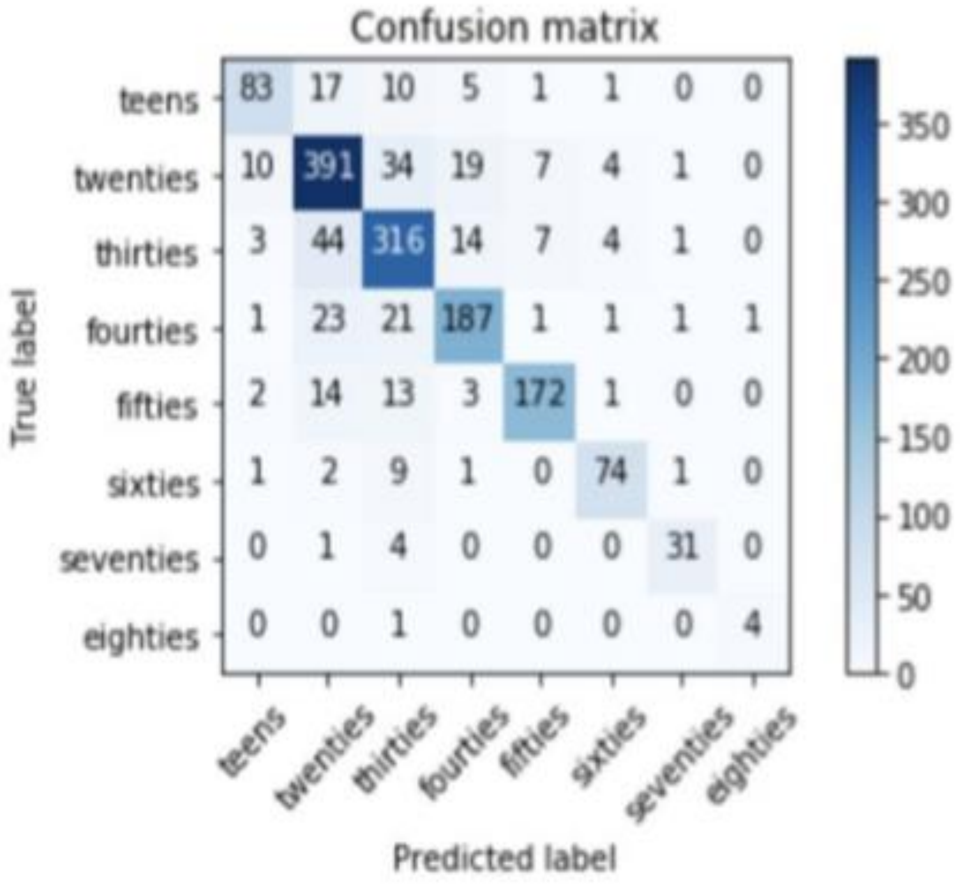
CHAPTER - 10
RESULTS AND DISCUSSIONS

Accuracy On Training Data :- 89%

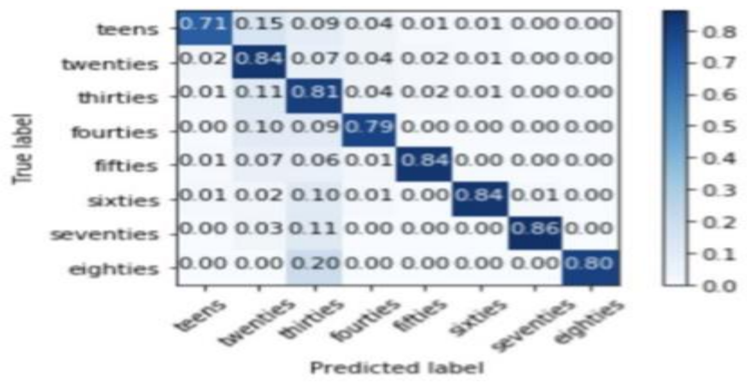
Accuracy On Test Data :- 84%

Highest Accuracy was found among Twenties as the data for the category was larger in number

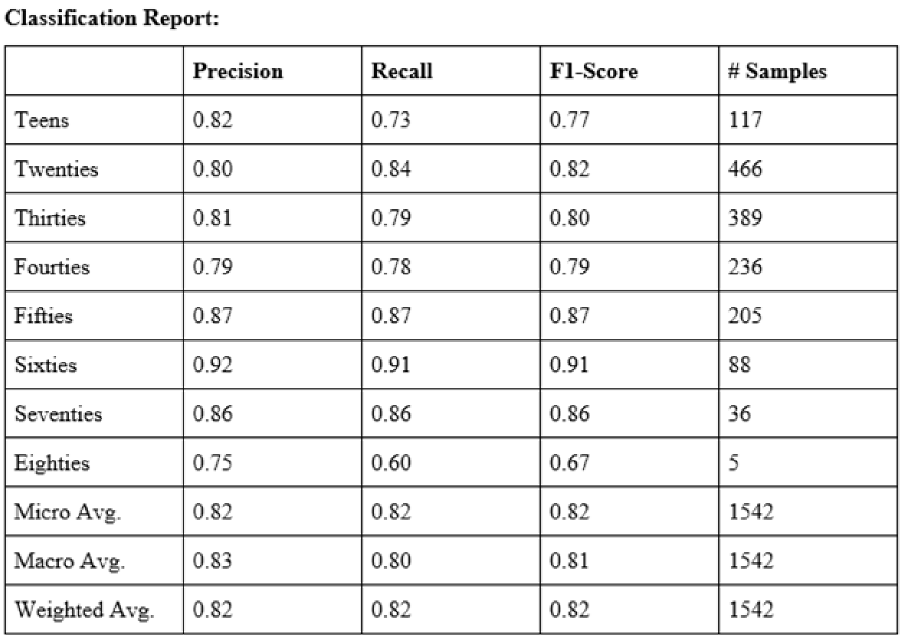
Confusion Matrix



True label \ Predicted label	teens	twenties	thirties	fourties	fifties	sixties	seventies	eighties
teens	83	17	10	5	1	1	0	0
twenties	10	391	34	19	7	4	1	0
thirties	3	44	316	14	7	4	1	0
fourties	1	23	21	187	1	1	1	1
fifties	2	14	13	3	172	1	0	0
sixties	1	2	9	1	0	74	1	0
seventies	0	1	4	0	0	0	31	0
eighties	0	0	1	0	0	0	0	4



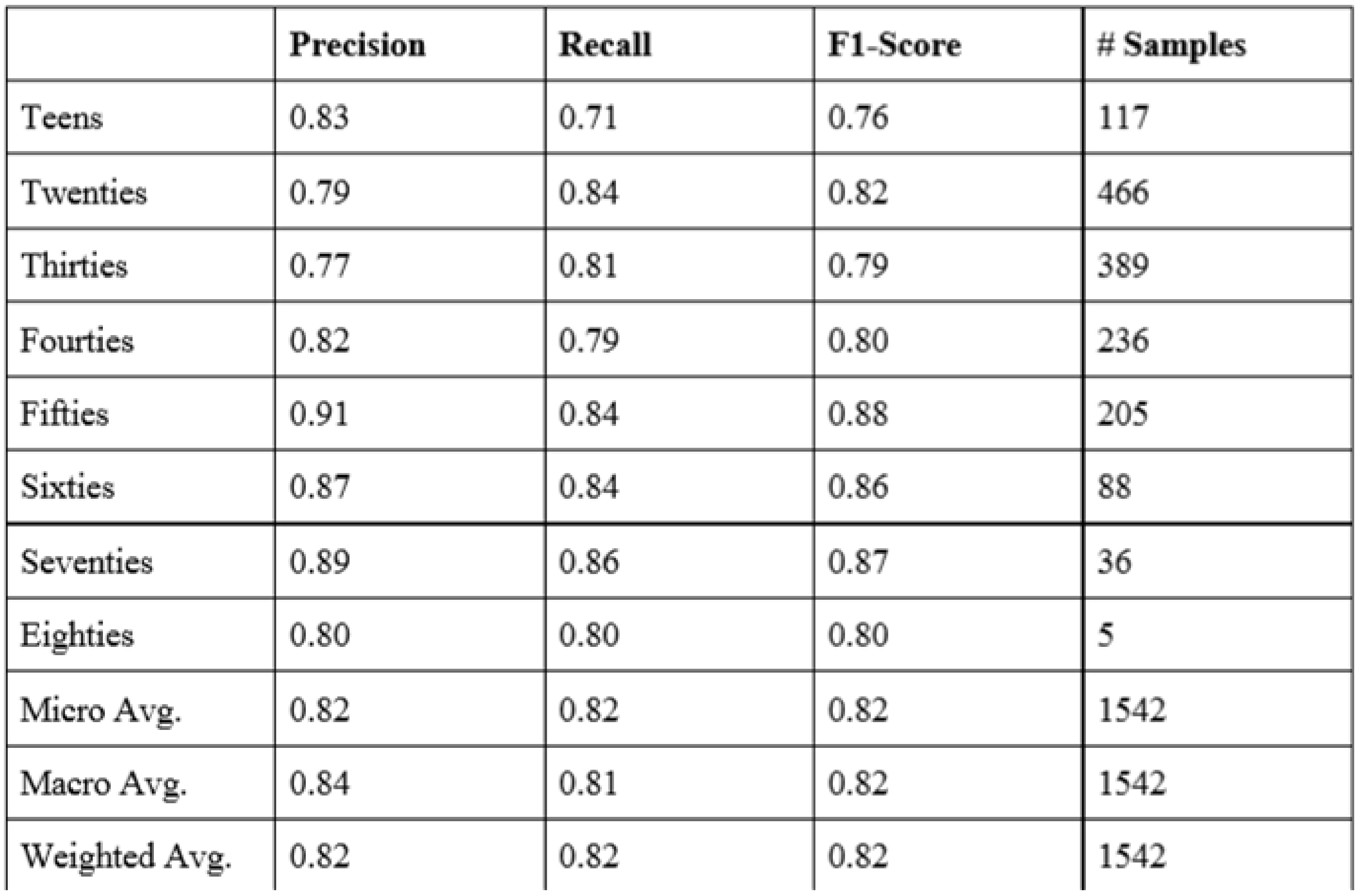
Classification Report



Classification Report:

	Precision	Recall	F1-Score	# Samples
Teens	0.82	0.73	0.77	117
Twenties	0.80	0.84	0.82	466
Thirties	0.81	0.79	0.80	389
Fourties	0.79	0.78	0.79	236
Fifties	0.87	0.87	0.87	205
Sixties	0.92	0.91	0.91	88
Seventies	0.86	0.86	0.86	36
Eighties	0.75	0.60	0.67	5
Micro Avg.	0.82	0.82	0.82	1542
Macro Avg.	0.83	0.80	0.81	1542
Weighted Avg.	0.82	0.82	0.82	1542

Classification Report:

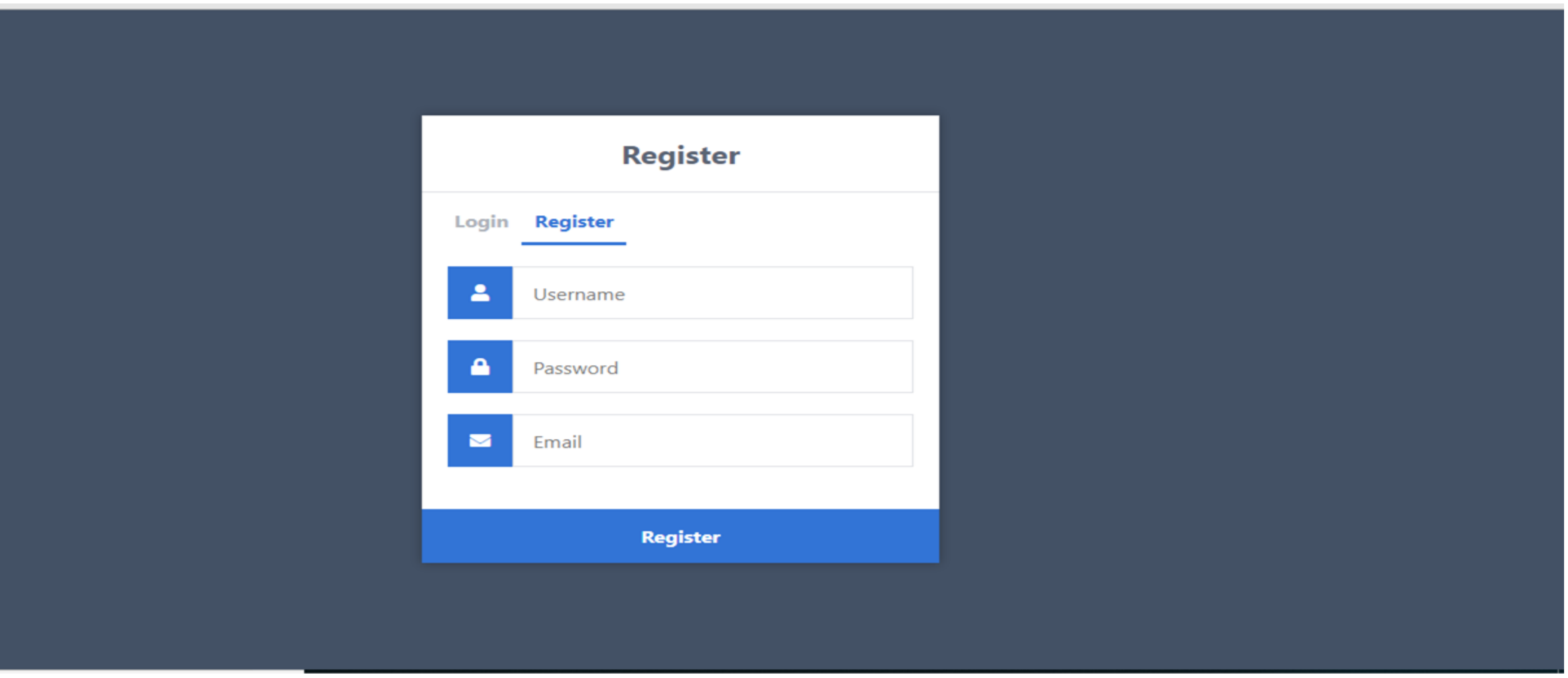


	Precision	Recall	F1-Score	# Samples
Teens	0.83	0.71	0.76	117
Twenties	0.79	0.84	0.82	466
Thirties	0.77	0.81	0.79	389
Fourties	0.82	0.79	0.80	236
Fifties	0.91	0.84	0.88	205
Sixties	0.87	0.84	0.86	88
Seventies	0.89	0.86	0.87	36
Eighties	0.80	0.80	0.80	5
Micro Avg.	0.82	0.82	0.82	1542
Macro Avg.	0.84	0.81	0.82	1542
Weighted Avg.	0.82	0.82	0.82	1542

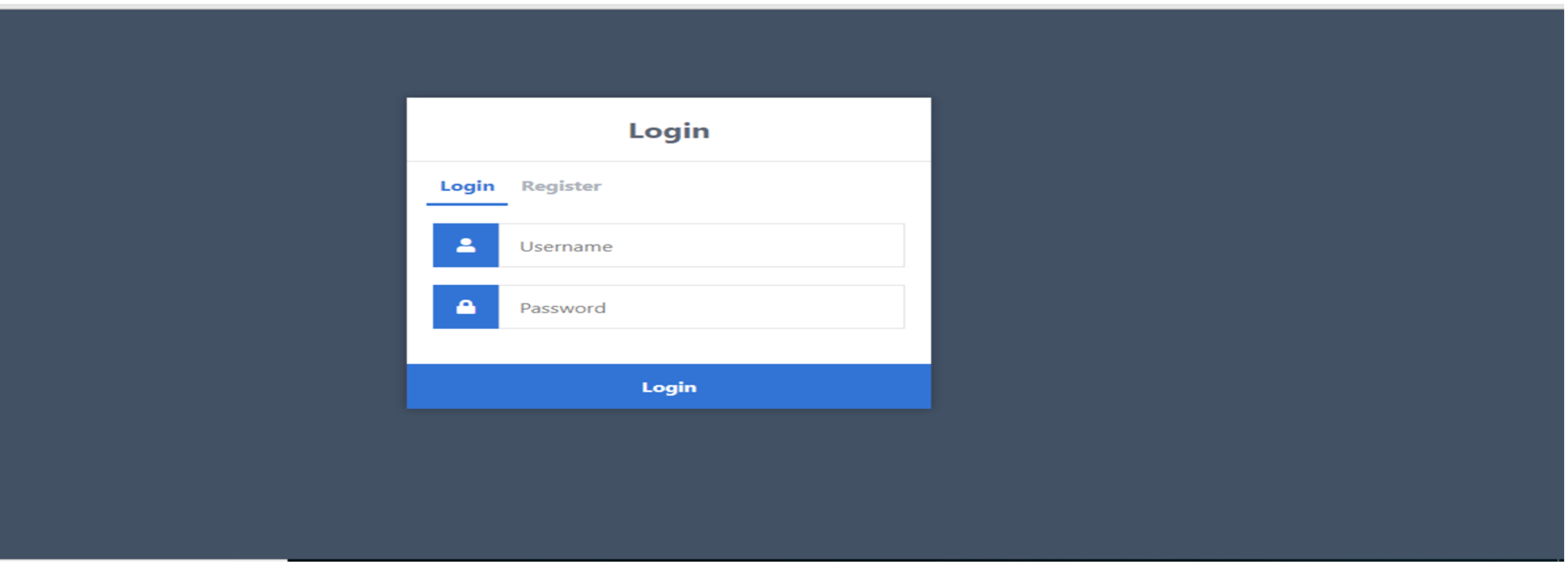
CHAPTER - 11

SNAPSHOTS

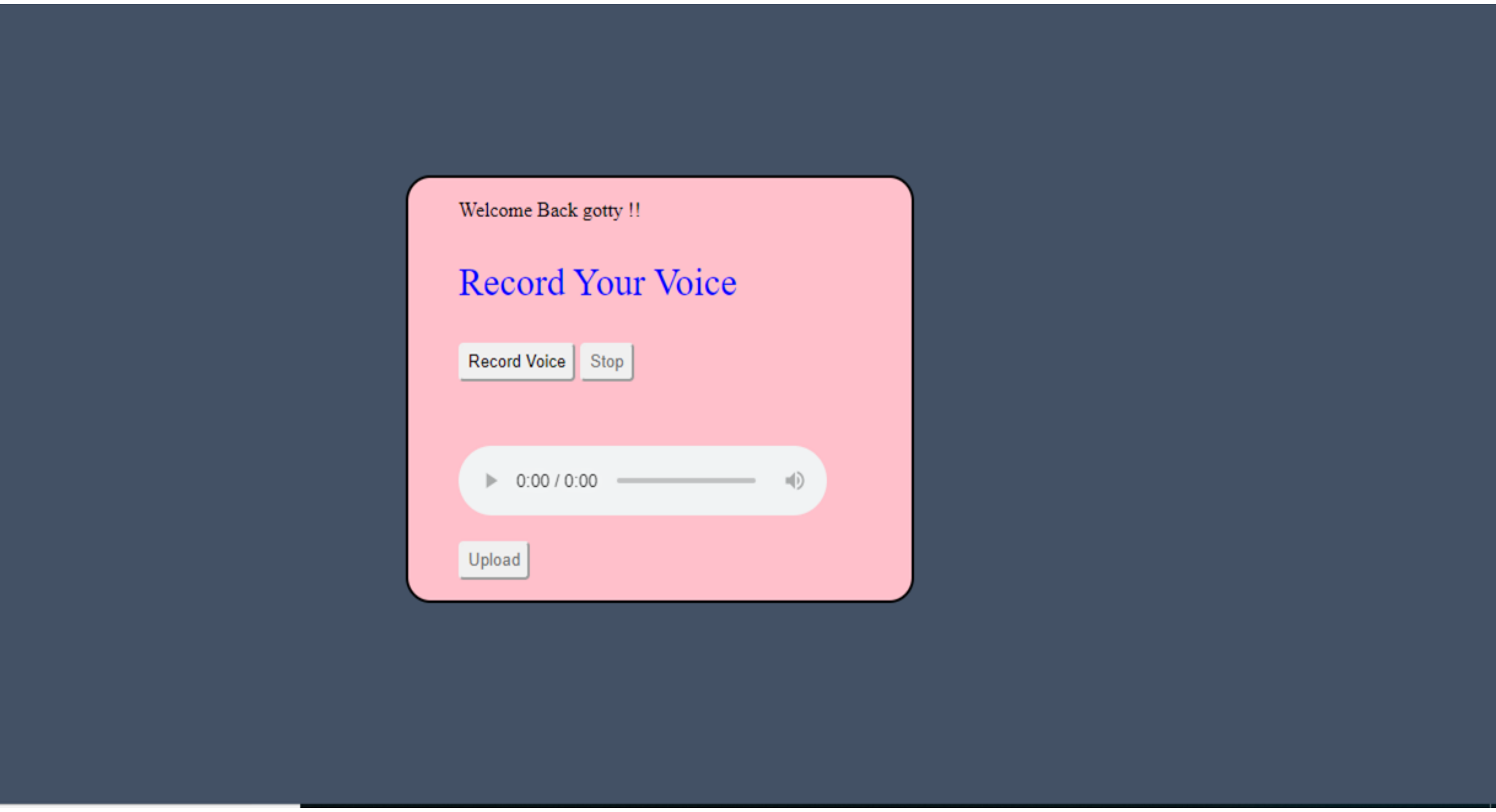
Register Page



Login Page

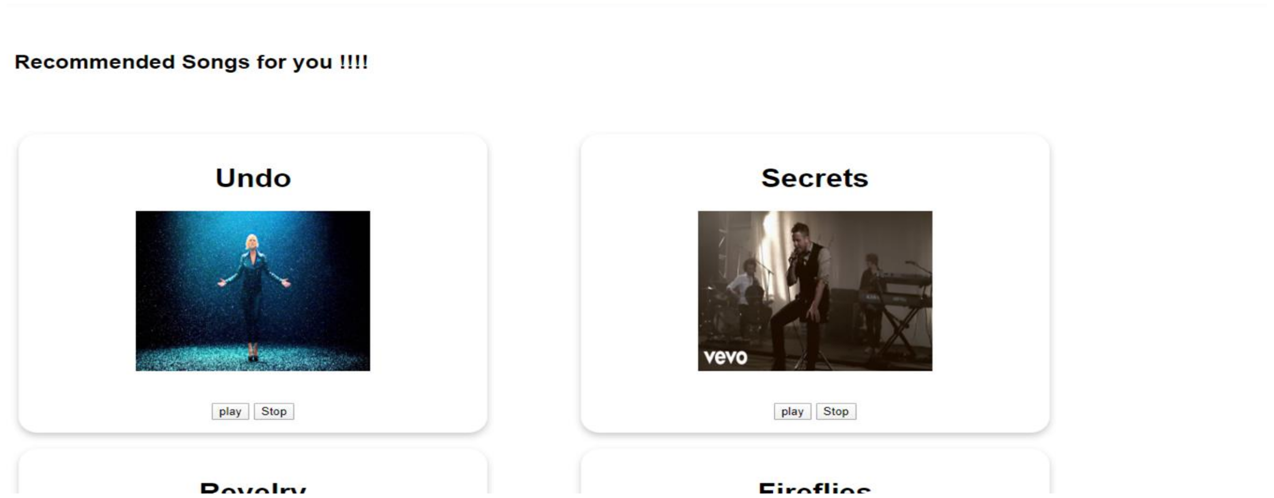


Audio Recording Page


Welcome Back gotty !!

Record Your Voice

Song Recommendation Page


Recommended Songs for you !!!!

CHAPTER – 12

Conclusions

With ever increasing devices being connected via IOT, need for personalization of results and optimizing it to cater to needs of the user increasing. A small thing such as an age can affect the search results vastly.

With ever increasing concern over privacy, people don't feel very good sharing personal information on website due to security reasons . At such cases it becomes difficult to optimize results for such users.

A system which could categorize one's age based on their voice could find it's use in many such cases. In the current project we have limited ourselves to a popularity based song recommendation based on the voice of the user.

We believe we have produced satisfactory results through our effort. We have achieved an accuracy of 84% on test data and further higher accuracy when the age group is in twenties, teens and thirties.

Working on this project gave us an immense learning experience and helped us learn concepts which we could use in our professional career ahead.

CHAPTER - 13
FUTURE ENHANCEMENTS

Connecting the project to an actually smart speaker or an voice activated device.

Implementation of the Audio Analysis to understand the demographic of a particular application that uses voice commands to run.

Integrating the song recommendation with premium Spotify APIs to provide a more streamlined song player.

Granularization of the age groups to provide further more age groups.

REFERENCES/ BIBLIOGRAPHY

[1] TITLE: Convolutional and recurrent Neural Network for Age and Gender detection from Speech

Source : IEEE – sept 2019

Authors : Héctor A. Sánchez-Hevia , Roberto Gil-Pita, Manuel Utrilla Manso

[2]TITLE : Using Voice to detect Age and Gender Based on Learning Generative Sparse Models

Source: International Journal of Engineering – sept 2018

Authors: S. Mavaddati

[3] Using Deep Neural Network Age Prediction

Source: Interspeech -2018

Authors: Pegah Ghahremani, Phani Sankar Nidadavolu, Nanxin Chen, Jesús Villalba, Daniel Povey, Sanjeev Khudanpur, Najim Dehak

APPENDIX A

DEFINITIONS, ACRONYMS AND ABBREVIATIONS

- CRNN - Convoluted Recurrent Neural Network
- FFMPEG - Fast Forward **MPEG** (Motion Picture Experts Group)
- MFCC – Mel-frequency cepstral coefficient