# Gender And Emotion Recognition Using Voice

Report submitted in partial fulfilment of the requirement for the degree of

B.Tech

In

Computer Science and Engineering

Under the Supervision of


**Ms. Pooja Mudgil**

**Assistant Professor, IT Department, BPIT**

by

**Preksha Singla 07520802713**

**Prakhar Agarwal 08320802713**

Department of Computer Science and Engineering
Bhagwan Parshuram Institute of Technology
PSP-4, Sec-17, Rohini, Delhi-89

May 2017

# DECLARATION

This is to certify that report titled "Gender and Emotion Recognition using Voice", is submitted by us in partial fulfilment of the requirement for the award of degree B.Tech in Computer Science and Engineering to BPIT, GGSIP University, Dwarka, Delhi. It comprises of our original work. The due acknowledgement has been made in the report for using others work.

**Date: 1st May 2017**

**Preksha Singla 07520802713**

**Prakhar Agarwal 08320802713**

# CERTIFICATE BY SUPERVISOR

This is to certify that the report title “Gender and Emotion Recognition using Voice" is submitted by Preksha Singla, Prakhar Agarwal to the department of CSE, Bhagwan Parshuram Institute of Technology, GGSIPU, Delhi . They are working under my supervision and this is to certify that the students are eligible for appearing in the first defense as per the criterion.

DATE Ms. Pooja Mudgil

# CERTIFICATE BY HOD

This is to certify that report titled “Gender and Emotion Recognition using Voice” is submitted by Preksha Singla and Prakhar Agarwal, under the guidance of Ms. Pooja Mudgil in partial fulfilment of the requirement for the award of degree B.Tech in Computer Science and Engineering to BPIT, GGSIP University, Dwarka, Delhi. The matter embodied in this Report is original and has been dully approved for the submission.

Date: Prof. Payal Pahwa

# ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my mentor teachers for their exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by them shall carry me a long way in the journey of life on which I am about to embark.

Lastly, I thank the Almighty, my parents and family for their constant encouragement without which this assignment would not have been possible

Preksha Singla

Prakhar Agarwal

# TABLE OF CONTENTS

# LIST OF FIGURES

# ABSTRACT

Identifying the gender and emotion of a speaker from speech has a variety of applications ranging from speech analytics to personalising human-machine interactions. While gender identification in previous works has explored the use of the statistical properties of the speaker's pitch features, in this project, we explore the impact of using acoustic features on identifying gender. In addition to gender we will also predict the emotion of speaker using the same acoustic values. We present a novel approach that models acoustic properties in the interest of identifying the speaker's gender and emotion with as little speech as possible. In this project we will investigate two datasets containing voice samples of over 3000 people for gender and over 1000 voice samples for emotions. Finally, we present various models for gender and emotion detection using the programming language R.


## Main Modules:

- User Interface for taking audio file as input and showing prediction for gender and emotion.
- R module for calculating acoustic properties of all sound files altogether and writing to a CSV file.
- R module for analysing CSV files and predicting gender and emotion for the sound clip uploaded.

# CHAPTER 1

# INTRODUCTION

## 1.1 ABOUT THE PROJECT

Determining a person's gender as male or female, based upon a sample of their voice seems to initially be an easy task. Often, the human ear can easily detect the difference between a male or female voice within the first few spoken words. However, designing a computer program to do this turns out to be a bit trickier. This project describes the design of a computer program to model acoustic analysis of voices and speech for determining gender and emotion. The model is constructed using more than 3,000 recorded samples of male and female voices, speech, and utterances plus over 1000 recorded samples for different emotions. The samples are processed using acoustic analysis and then applied to an artificial intelligence/machine learning algorithm to learn gender-specific traits. The resulting program achieves 89% accuracy on the test set.

## 1.2 EXISTING SYSTEM (problems):

People can identify gender and emotions of other people easily just by listening to their voice but training a computer program to this is a difficult task. Building a computer program to identify gender and emotion can be used in various technologies for making great user experiences. Voice recognition can be used in artificial intelligent systems. In general identification of a speaker gender is important for increasingly natural and personalised dialogue systems.

## 1.3 PROPOSED SYSTEM:

The proposed system provides the facility to determine a person's gender and emotion from his/ her speech. The system is provided with 3000+ voice samples for gender recognition and 800+ voice samples for emotion recognition. The system extracts the features from the voice samples and stores them in 'CSV' files. These files are used to build models for prediction of gender and emotion. The gender and emotion of new voice sample received is predicted using these models.

## 1.4 FEATURES OF PROPOSED SYSTEM:

*1.4.1 FUNCTIONAL CAPABILITIES:* The main aim of this project is to determine the speakers gender and emotions based upon the acoustic parameters calculated.

The user is required to upload a speech file in '.wav' format for analysis. There is no need of internet for running this project.

*1.4.2. PERFORMANCE LEVEL:* The scope of this project gives immense opportunity for reduced labour and increased performance.

*1.4.3. DATA STORAGE:* The features extracted from the .wav file are stored in a cvs file.

*1.4.4. SAFETY:* No loss occurs in the system.

*1.4.5. RELIABILITY:* We assure that the project is completely reliable. The system predicts the gender and emotion correctly in most of the cases.

# CHAPTER 2

# RELATED WORK

## 2.1. Voice

Voice (or vocalisation) is the sound produced by humans and other vertebrates using the lungs and the vocal folds in the larynx, or voice box. Voice is not always produced as speech, however. Your voice is as unique as your fingerprint. It helps define your personality, mood, and health.

## 2.2. R Programming Language[3]

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

## 2.3. Shiny[4]

Shiny is an open source R package that provides an elegant and powerful web framework for building web applications using R. Shiny helps you turn your analyses into interactive web applications without requiring HTML, CSS, or JavaScript knowledge.

## 2.4. CSV (Comma Separated Value)

In computing, a comma-separated values (CSV) file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.

## 2.5. tuneR[2]

Analyse music and speech, extract features like MFCCs, handle wave files and their representation in various ways, read mp3, read midi, perform steps of a transcription, ... Also contains functions ported from the 'rastamat' 'Matlab' package.

## 2.6. seewave[1]

Functions for analysing, manipulating, displaying, editing and synthesizing time waves (particularly sound). This package processes time analysis (oscillograms and envelopes), spectral content, resonance quality factor, entropy, cross correlation and autocorrelation, zero-crossing, dominant frequency, analytic signal, frequency coherence, 2D and 3D spectrograms and many other analyses.

## 2.7. R Random Forest[6]

In the random forest approach, a large number of decision trees are created. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output. A new observation is fed into all the trees and taking a majority vote for each classification model.

The R package "randomForest" is used to create random forests**.**

## 2.8. CART Model

When utilising an algorithm such as logistic regression, it can be difficult to determine which exact properties indicate a target gender of male or female. We could guess that it likely one of the statistically significant features, but ultimately this decision breakdown is masked within the model. To gain an understanding of a trained model, we can apply a classification and regression tree model (CART) to our dataset to determine how these properties might correspond to a gender classification of male or female.

## 2.9. SVM Model

Our next model is a support vector machine, tuned with the best values for cost and gamma. To determine the best fit for an SVM model, the model was initially run with default parameters. A plot of the SVM error rate is then printed, with the darkest shades of blue indicating the best (ie., lowest) error rates. This is the best place to choose a cost and gamma value. You can fine-tune the SVM by narrowing in on the darkest blue range and performing further tuning. This essentially focuses in on the section, yielding a finer value for cost and gamma, and thus, a lower error rate and higher accuracy. The following performance images show how this progresses.

# CHAPTER 3

# System Analysis and Design

## 3.1. SOFTWARE REQUIREMENT SPECIFICATION (SRS)

*3.1.1. Introduction:*

The following subsections of the SRS document provide an overview of the entire SRS.

*3.1.1.1. Purpose:* The purpose of the project is to predict gender and emotions of a person using a voice sample.

*3.1.1.2. Scope:* The application will train a computer program to make predictions on gender and emotions by taking input any voice sample or utterance of a person.

*3.1.1.3. Benefits:* The proposed application provides automated recognition, reduces the manual work and saves user's time and money.

*3.1.1.4. Overview:* The rest of this SRS document describes the various systems requirements, interfaces, features and functionalities in detail.

*3.1.2. Overall Description:*

The Automatic Irrigation system has two main parts:

- Software

- Hardware

In Software part, we have made an R application that has 3 main modules namely, module for creating CSV from voice samples,module for analysis of CSV file and predicting gender and emotion and a module for showing output using shiny. We have taken a dataset of over 3000 voice samples for gender recognition and downloaded over 1000 voice samples for emotion recognition.

*3.1.3. Specific Requirements:*

This section provides software requirements to a level of detail sufficient to enable designers to design the system and testers to test the system.

*3.1.3.1. FRAMEWORK*

- R Studio: R Studio is a free and open-source integrated development environment (IDE) for R, a programming language for statistical computing and graphics. R Studio is written in the C++ programming language and uses the Qt framework for its graphical user interface.

*3.1.3.2. BACKEND (DATABASE)*

- CSV Files: In computing, a comma-separated values (CSV) file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.

*3.1.3.3. SERVER*

- Shiny: Shiny is an open source R package that provides an elegant and powerful we framework for building web applications using R. Shiny helps you turn your analyses into interactive web applications without requiring HTML, CSS, or JavaScript knowledge.

*3.1.3.4. HARDWARE*

Hardware will consist of a normal computer which can run R Studio and has a support for using a microphone.

*3.1.3.5. Product Functions:*

*SOFTWARE:*

A summary of the major functions that the application will perform:

a. Provide functionality to read voice samples and extract acoustic properties and write to a CSV file.

b. Provide the functionality to model the data using different models.

c. Provide functionality to take input through user a voice sample and show output of predicted gender and emotion.

*HARDWARE:*

A summary of the major functions that the application will perform:

a. A microphone for taking input voice sample of a user.

*3.1.3.6. User Characteristics:*

-Educational level: Users should be comfortable with English language.

-Experience: Users should have

a. A computer with a microphone to record voice.

*3.1.4. NON FUNCTIONAL REQUIREMENTS*

They are the quality requirements that stipulate how well software does what it has to do.

*3.1.4.1. Performance*

No. of terminals to be supported is dependent on the server that we will use at the time of deployment. The server used should provide good performance and ability to manage performance with techniques such as support for caching.

*3.1.4.2. Reliability*

It means the extent to which program performs with required precision. The application developed should be extremely reliable and secure

*3.1.4.3. Usability*

The application should be user friendly and should require least effort to operate.

*3.1.4.4. Portability*

The application is made using R Framework, thus can be transported to any other device with minimum effort.

*3.1.4.5. Flexibility*

It is effort required to modify operational program. The whole application should be made using independent modules so that any changes done in 1 module should not affect the other one and new modules can be added easily to increase functionality.

*3.1.5. FEASIBILITY STUDY:*

*3.1.5.1. What are the user's demonstrable needs?*

User needs an intelligent system, which will remove all the above-mentioned problems that, the user is facing. A system which will reduce provide ease of work and flexibility.

*3.1.5.2. How can the problem be redefined?*

We proposed our perception of the system, in accordance with the problems of existing system by making a full layout of the system on paper. We tallied the problems and needs by existing system and requirements. In feasibility study phase we had undergone through various steps, which are described as under:

*3.1.5.3. How feasible is the system proposed?*

This was analysed by comparing the following factors with both the existing system and proposed system.

*3.1.5.4. Cost:*

The cost required in the proposed system is comparatively more to the existing system.

*3.1.5.5. Effort:*

Compared to the existing system the proposed system will provide a better environment in which there will be ease of work and the effort required will be comparatively less than the existing system.

*3.1.5.6. Time:*

Also the time required will be comparatively very less than in the existing system.

*3.1.5.7. Labor:*

The new system will require quite less labour.

## 3.2. DIAGRAMS

### *3.2.1 USE CASE DIAGRAM*

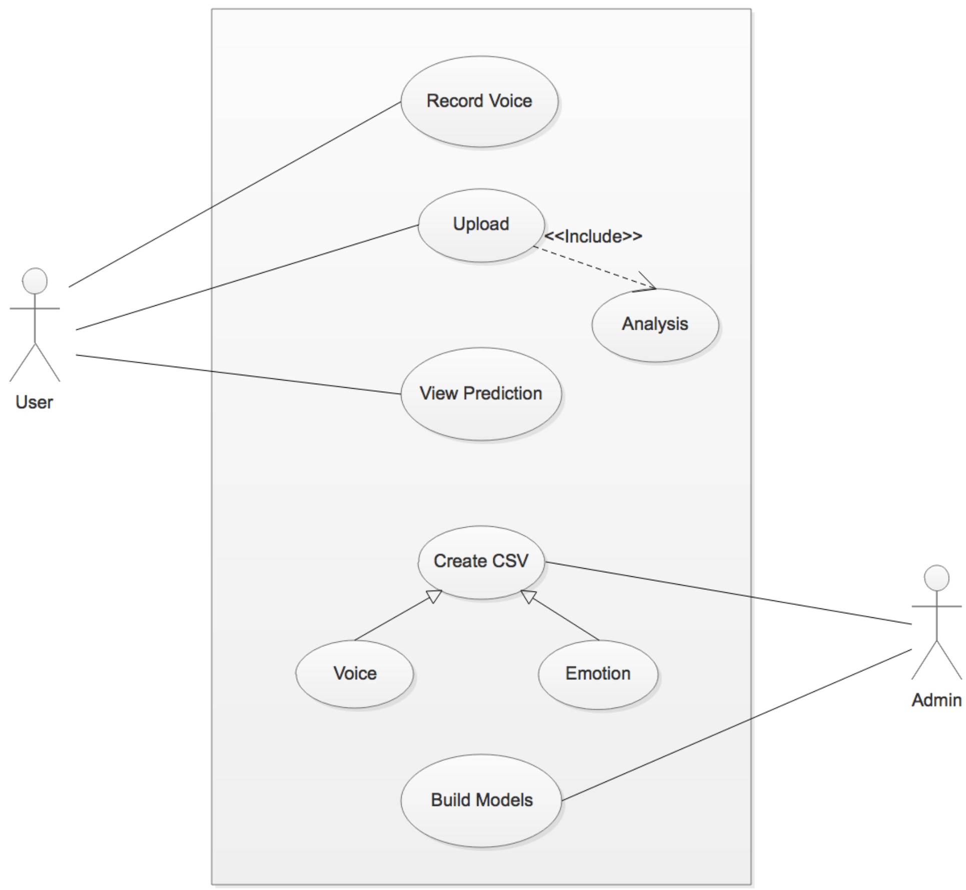


*Fig 3.1 USECASE DIAGRAM*

*3.2.2 DFD LEVEL - 0*

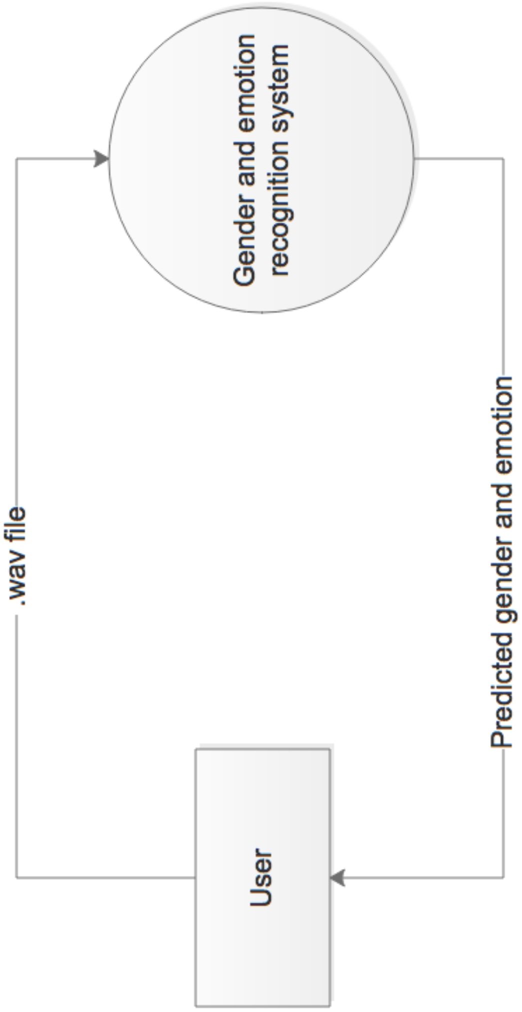


*Fig 3.2 DFD LEVEL 0*

### *3.2.3. DFD LEVEL-1*

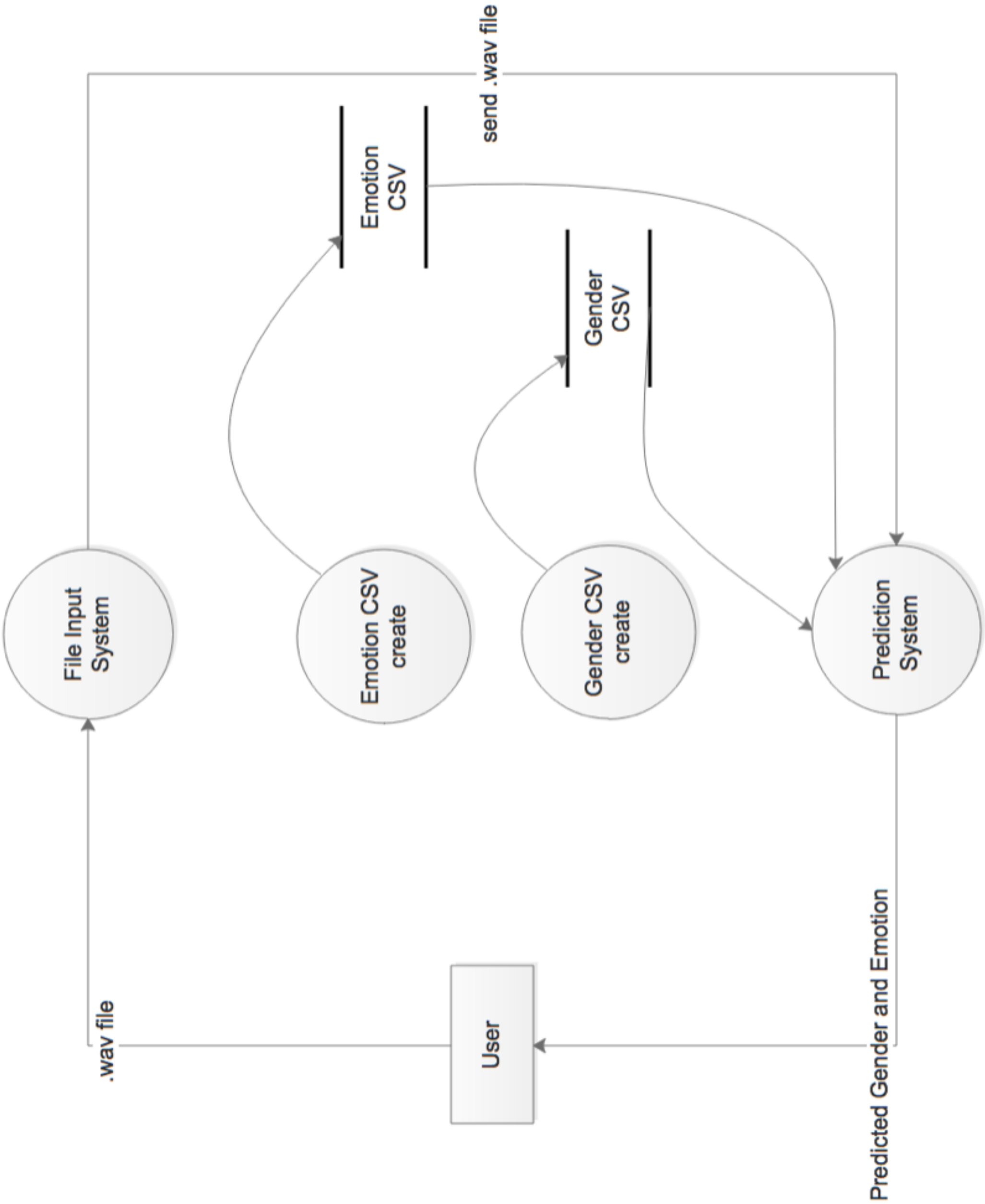


*Fig 3.3 DFD LEVEL 1*

*3.2.4. FLOW CHART*

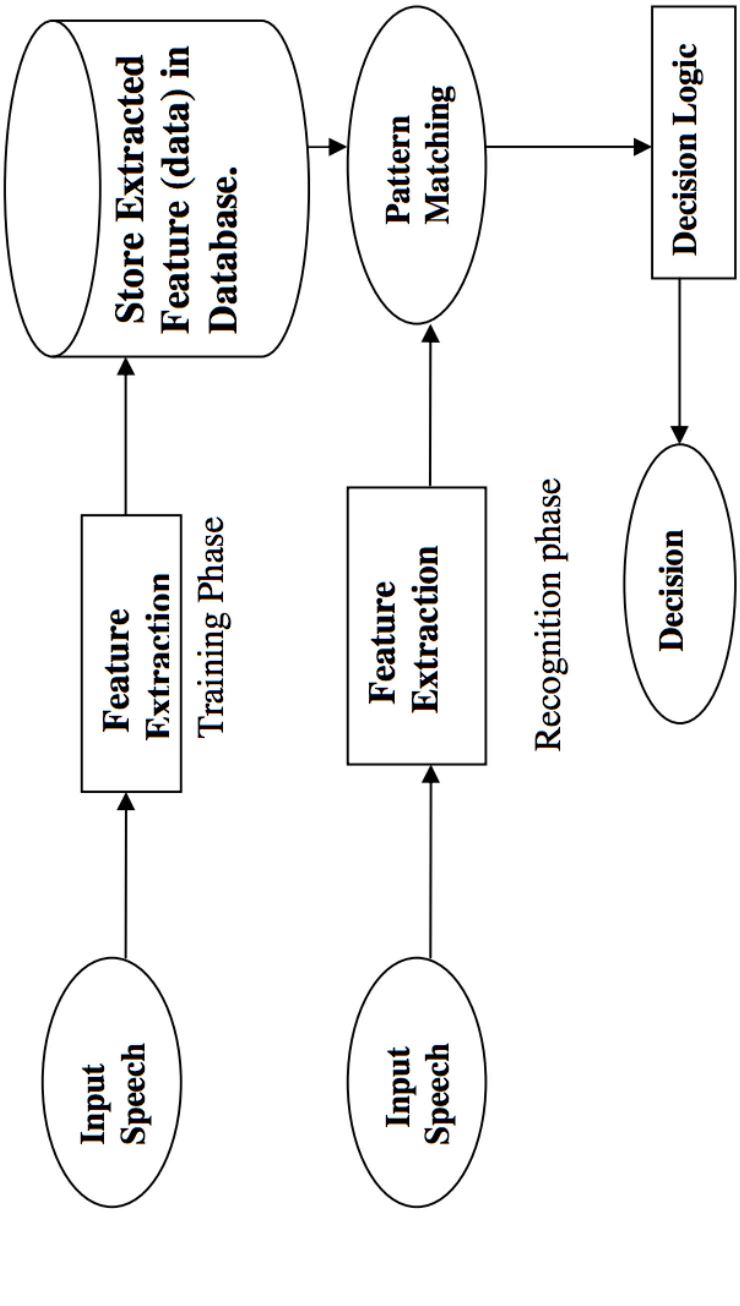


*Fig 3.4 Flow Chart*

# CHAPTER 4
# PROPOSED WORK

This project involves following two functionalities

1. Gender Recognition
2. Emotion Recognition

## 4.1. GENDER RECOGNITION

This feature will take input a voice sample from user and analyse it to predict the gender of the user. This feature will train from a dataset of over 3000 voice samples which is made by extracting acoustic properties of sample through seewave package in R and storing in a CSV file. Following are the various steps of implementation of this functionality.

### *4.1.1. Recording Voice Samples*

In this phase, over 3000 voice samples are collected from users either by recording them or by downloading them from internet and storing them separately for male and female voices in folders- Male and Female.

### *4.1.2. Extracting Acoustic Properties*

In this phase various acoustic properties are extracted from all the voice samples in one go and storing the values in a CSV file.

Acoustic Properties Measured—

- *duration*: length of signal
- *meanfreq*: mean frequency (in kHz)
- *sd*: standard deviation of frequency
- *median*: median frequency (in kHz)

- *Q25*: first quantile (in kHz)
- *Q75*: third quantile (in kHz)
- *IQR*: interquantile range (in kHz)
- *skew*: skewness (see note in specprop description)
- *kurt*: kurtosis (see note in specprop description)
- *sp*.ent: spectral entropy
- *sfm*: spectral flatness
- *mode*: mode frequency
- *centroid*: frequency centroid (see specprop)
- *peakf*: peak frequency (frequency with highest energy)
- *meanfun*: average of fundamental frequency measured across acoustic signal
- *minfun*: minimum fundamental frequency measured across acoustic signal
- *maxfun*: maximum fundamental frequency measured across acoustic signal
- *meandom*: average of dominant frequency measured across acoustic signal
- *mindom*: minimum of dominant frequency measured across acoustic signal
- *maxdom*: maximum of dominant frequency measured across acoustic signal
- *dfrange*: range of dominant frequency measured across acoustic signal
- *modindx*: modulation index. Calculated as the accumulated absolute difference

### *4.1.3. Creating CSV File*

| meanfreq | sd | median | Q25 | Q75 | IQR | skew | kurt | sp.ent | sfm | mode | centroid | meanfun | minfun | maxfun | meandom | mindom | maxdom | dfrange | modindx | label |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.0597809849598081 | 0.0642412677031359 | 0.03202691337258 2 | 0.0150714886459209 | 0.0901934398654331 | 0.0751219512195122 | 12.86346 | 274.4029 | 0.893369416700807 | 0.491917766397811 | 0 | 0.0597809849598081 | 0.084279106440321 | 0.0157016683022571 | 0.275862068965517 | 0.007812 | 0.007812 | 0.007812 | 0 | 0 | male |
| 0.066008740387572 | 0.0673100287952527 | 0.040228734810579 | 0.0194138670478914 | 0.0926661901358113 | 0.0732523230879199 | 22.42328 | 634.6138 | 0.892193242265734 | 0.513723842537073 | 0 | 0.066008740387572 | 0.107936553670454 | 0.0158259149357072 | 0.25 | 0.00901442307692308 | 0.007812 | 0.054687 | 0.046875 | 0.0526315789473684 | male |
| 0.0773155026958227 | 0.0838294209445061 | 0.0367184586699814 | 0.00870105655686762 | 0.131908017402113 | 0.123206960845246 | 30.75715 | 1,024.927 | 0.846389091878782 | 0.478904979116727 | 0 | 0.0773155026958227 | 0.0987062615673936 | 0.0156555772994129 | 0.271186440677966 | 0.0079900568181 82 | 0.007812 | 0.015625 | 0.007812 | 0.0465116279069767 | male |
| 0.151228091724635 | 0.0721105872627985 | 0.158011187072716 | 0.0965817277812306 | 0.207955251709136 | 0.111373523927906 | 1.232831 | 4.177296 | 0.963322461535984 | 0.727231798861951 | 0.0838781852082039 | 0.151228091724635 | 0.0889648485504597 | 0.0177975528364 85 | 0.25 | 0.201497395833333 | 0.007812 | 0.5625 | 0.554687 | 0.247119078104994 | male |
| 0.135120387296677 | 0.0791461004935869 | 0.124656228727025 | 0.0787202178352621 | 0.206044928522805 | 0.127324710687543 | 1.101173 | 4.333713 | 0.971955076212905 | 0.783568057553871 | 0.104261 | 0.135120387296677 | 0.106397844620363 | 0.0169312169312169 | 0.266666666666667 | 0.712812 | 0.007812 | 5.484375 | 5.476562 | 0.208273894436519 | male |
| 0.132786407306188 | 0.0795568659729794 | 0.119089848308051 | 0.0679579929988331 | 0.209591598599767 | 0.141633605600933 | 1.932562 | 8.308895 | 0.963181 | 0.738307 | 0.112555425904317 | 0.132786407306188 | 0.110131920122721 | 0.0171122994652406 | 0.253968253968254 | 0.298221982758621 | 0.007812 | 2.726562 | 2.71875 | 0.125159642401022 | male |

*FIG 4.1 Gender CSV*

All the acoustic properties extracted for all voice samples are stored in a CSV file.

*4.1.4. Training With Models*

In this phase we train the program on this data set using various models and predict gender on a test set.

Various models used are—

- Random Forest
- CART Model
- SVM Model

*4.1.5 Shiny APP*

In this phase we create web application using Shiny to take input voice sample from user and showing the predicted value of gender using the above models

## 4.2. EMOTION RECOGNITION

This feature will take input a voice sample from user and analyse it to predict the emotion of the user. This feature will train from a dataset of over 1000 voice samples which is made by extracting acoustic properties of sample through seewave package in R and storing in a CSV file. Following are the various steps of implementation of this functionality.

*4.2.1. Recording Voice Samples*

In this phase, over 1000 voice samples are collected from users either by recording them or by downloading them from internet and storing them separately for emotions -

- Neutral
- Angry
- Sad
- Fear

### *4.2.2. Extracting Acoustic Properties*

In this phase various acoustic properties are extracted from all the voice samples in one go and storing the values in a CSV file.

Acoustic Properties Measured—

- *duration*: length of signal
- *meanfreq*: mean frequency (in kHz)
- *sd*: standard deviation of frequency
- *median*: median frequency (in kHz)
- *Q25*: first quantile (in kHz)
- *Q75*: third quantile (in kHz)
- *IQR*: interquantile range (in kHz)
- *skew*: skewness (see note in specprop description)
- *kurt*: kurtosis (see note in specprop description)
- *sp*.ent: spectral entropy
- *sfm*: spectral flatness
- *mode*: mode frequency
- *centroid*: frequency centroid (see specprop)
- *peakf*: peak frequency (frequency with highest energy)
- *meanfun*: average of fundamental frequency measured across acoustic signal
- *minfun*: minimum fundamental frequency measured across acoustic signal
- *maxfun*: maximum fundamental frequency measured across acoustic signal
- *meandom*: average of dominant frequency measured across acoustic signal
- *mindom*: minimum of dominant frequency measured across acoustic signal
- *maxdom*: maximum of dominant frequency measured across acoustic signal
- *dfrange*: range of dominant frequency measured across acoustic signal

- *modindx*: modulation index. Calculated as the accumulated absolute difference

### *4.2.3. Creating CSV File*

All the acoustic properties extracted for all voice samples are stored in a CSV file.

| meanfreq | sd | median | Q25 | Q75 | IQR | skew | kurt | sp.ent | sfm | mode | centroid | meanfun | minfun | maxfun | meandom | mindom | maxdom | dfrange | modindx | label |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.187383870066221 | 0.0469853445774683 | 0.199933110367893 | 0.193377926421405 | 0.205083612040134 | 0.011705685618 7291 | 5.743882 | 43.30470 | 0.776211 | 0.272873522821225 | 0.199933110367893 | 0.187383870066221 | 0.188044346373421 | 0.0362225519287834 | 0.265369565217391 | 1.468654 | 0 | 10.37118 | 10.37118 | 0.128113026819923 | neutral |
| 0.174112385700404 | 0.0552044792425174 | 0.190329 | 0.185494505494505 | 0.198241758241758 | 0.0127472527472528 | 5.743627 | 43.27521 | 0.786805505242834 | 0.304910581770547 | 0.189890 | 0.174112385700404 | 0.175665340434886 | 0.0247857868020305 | 0.256989 | 1.946315 | 0.190734 | 8.773781 | 8.583046 | 0.0722800925925926 | neutral |
| 0.192817169909523 | 0.0470657844827469 | 0.2044 | 0.199266666666667 | 0.209066666666667 | 0.00979999999999998 | 4.896726 | 28.83642 | 0.751075544565975 | 0.248368159314965 | 0.2058 | 0.192817169909523 | 0.182474086649909 | 0.0345807365439093 | 0.230320754716981 | 1.974895 | 0.035762 | 9.036041 | 9.000278 | 0.0993377483443709 | neutral |
| 0.172378824415843 | 0.0524192429465304 | 0.186501766784452 | 0.182049469964664 | 0.191448763250883 | 0.0093992932862190 6 | 6.218212 | 49.00626 | 0.772328871141711 | 0.299858 | 0.185512367491166 | 0.172378824415843 | 0.157103995890483 | 0.0260833333333333 | 0.246606060606061 | 2.041065 | 0.178813 | 9.107566 | 8.928752 | 0.128170894526035 | neutral |
| 0.179164510452416 | 0.0540929617993851 | 0.192820512820513 | 0.187692307692308 | 0.201538461538462 | 0.0138461538461538 | 5.137214 | 36.33489 | 0.780520216220941 | 0.251559858641525 | 0.191282051282051 | 0.179164510452416 | 0.177343949331316 | 0.025431 | 0.241722772277228 | 1.423769 | 0 | 9.131408 | 9.131408 | 0.135176833610254 | neutral |
| 0.170448750021394 | 0.0509054710366123 | 0.184044943820225 | 0.178801498127341 | 0.187715355805243 | 0.0089138576779026 3 | 7.952093 | 79.84042 | 0.743390150407703 | 0.279726381018614 | 0.183520599250936 | 0.170448750021394 | 0.164328305653116 | 0.0261672025723473 | 0.230320754716981 | 1.885669 | 0.178813 | 9.238696 | 9.059882 | 0.128258145363409 | neutral |

*FIG 4.2 Emotion CSV*

### *4.2.4. Training With Models*

In this phase we train the program on this data set using various models and predict gender on a test set.

Various models used are—

- Random Forest
- CART Model
- SVM Model

### 4.2.5 Shiny APP

In this phase we create web application using Shiny to take input voice sample from user and showing the predicted value of gender using the above models

# CHAPTER 5
# RESULTS

## 5.1. GENDER PREDICTION

*5.1.1. Random Forest Model Prediction*

```
Console ~/Documents/GenderVoice/
  female     464    11
  male        10   465
> set.seed(777)
> genderForest <- randomForest(label ~ ., data=train)
>
> predictForest <- predict(genderForest, newdata=train)
> table(train$label, predictForest)
        predictForest
         female male
  female   1109    0
  male        0 1110
>
> predictForest <- predict(genderForest, newdata=test)
> table(test$label, predictForest)
        predictForest
         female male
  female    464   11
  male        8  467
>
```

*FIG 5.1 Random Forest Prediction for Gender*

*5.1.2. SVM Model Prediction*

```
Console ~/Documents/GenderVoice/
> set.seed(777)
> genderSvm <- svm(label ~ ., data=train, gamma=0.21, cost=8)
>
> predictSvm <- predict(genderSvm, train)
> table(predictSvm, train$label)

predictSvm female male
    female   1109    0
    male        0 1110
>
> predictSvm <- predict(genderSvm, test)
> table(predictSvm, test$label)

predictSvm female male
    female    465    7
    male       10  468
> predictForest <- predict(genderForest, newdata=train)
> table(train$label, predictForest)
        predictForest
         female male
```

*FIG 5.2 SVM Prediction for Gender*

*5.1.3 CART Model Prediction*

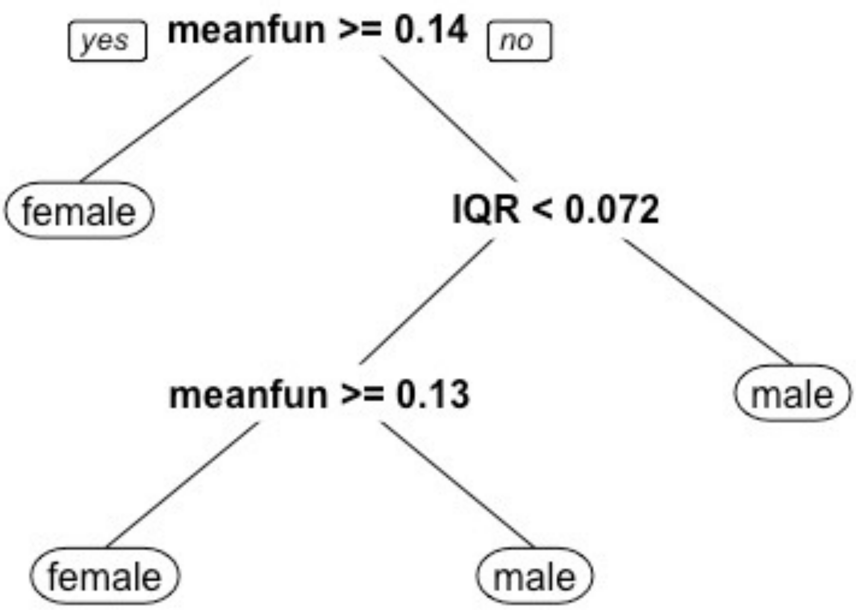


*FIG 5.3 CART Prediction for Gender*

## 5.2. EMOTION PREDICTION

### *5.2.1. Random Forest Model Prediction*

```
Console ~/Documents/GenderVoice/fear/
>
> predictForest <- predict(genderForest, newdata=train)
> table(train$label, predictForest)
         predictForest
          neutral angry sad fear
  neutral      35     0   0    0
  angry         0    35   0    0
  sad           0     0  35    0
  fear          0     0   0   35
>
> predictForest <- predict(genderForest, newdata=test)
> table(test$label, predictForest)
         predictForest
          neutral angry sad fear
  neutral      15     0   0    0
  angry         0    15   0    0
  sad           0     0  15    0
  fear          0     0   0   15
>
```

FIG 5.4 Random Forest Prediction for Emotion

### *5.2.2. SVM Model Prediction(ON)*

```
Console ~/Documents/GenderVoice/fear/
> genderSvm <- svm(label ~ ., data=train, gamma=0.21, cost=8)
>
> predictSvm <- predict(genderSvm, train)
> table(predictSvm, train$label)

predictSvm neutral angry sad fear
   neutral      35     0   0    0
   angry         0    35   0    0
   sad           0     0  35    0
   fear          0     0   0   35
>
> predictSvm <- predict(genderSvm, test)
> table(predictSvm, test$label)

predictSvm neutral angry sad fear
   neutral      14     0   0    0
   angry         0    15   0    0
   sad           0     0  15    0
   fear          1     0   0   15
>
```

*FIG 5.5 SVM Model Prediction for Emotion*

### *5.2.3. CART Model Prediction*

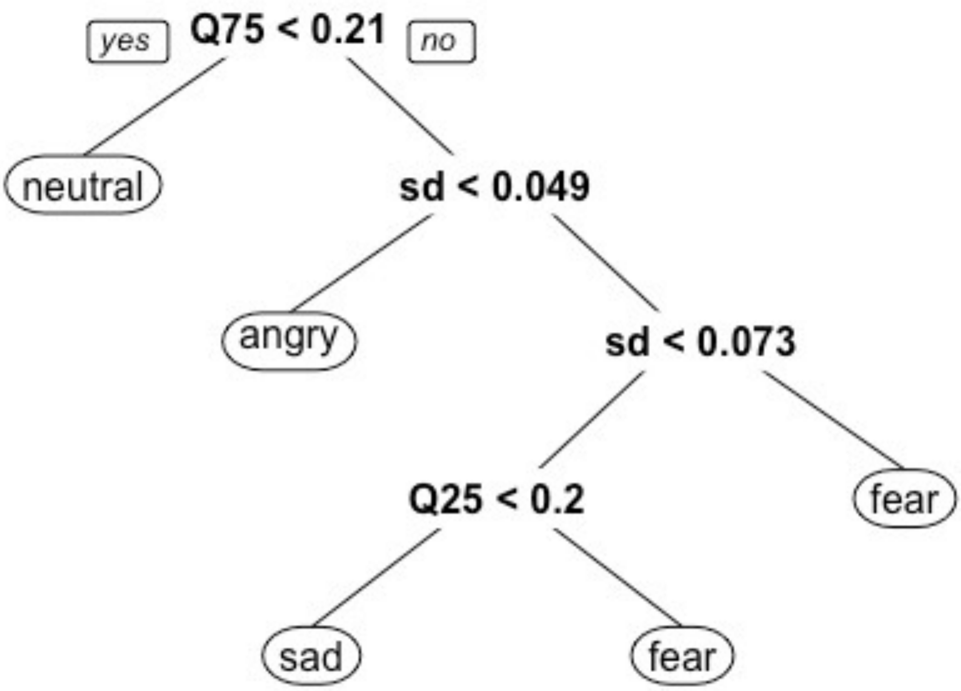


*FIG 5.6 CART Model Prediction for Emotion*

## 5.3. Shiny App UI Output

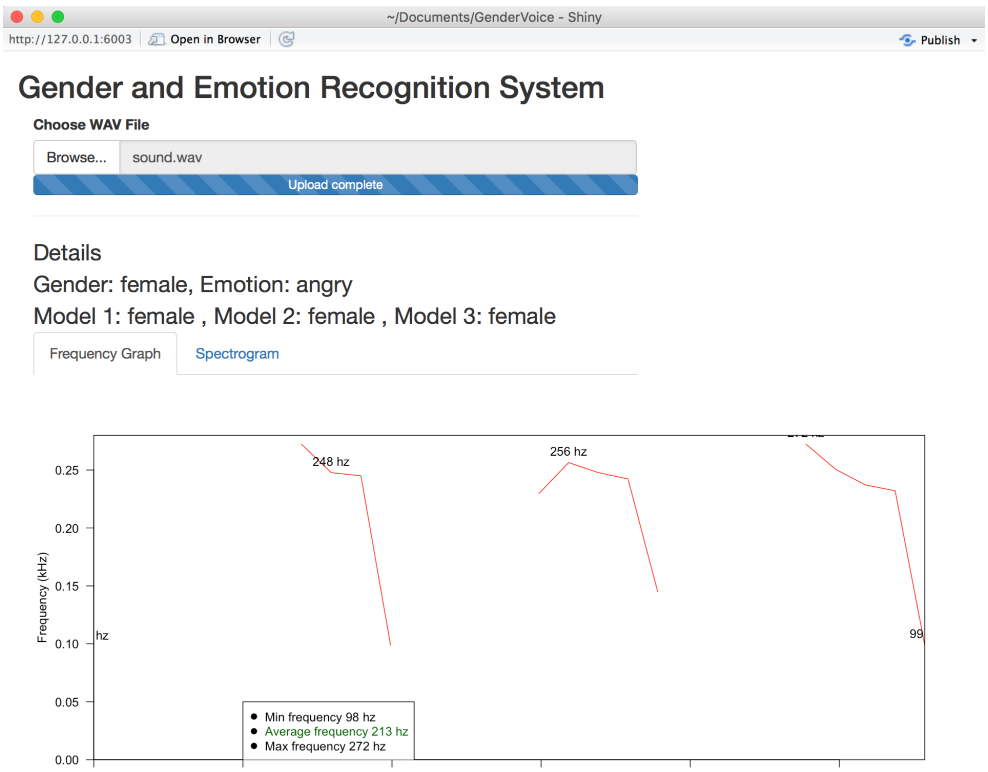


*FIG 5.7 Shiny App Output 1*

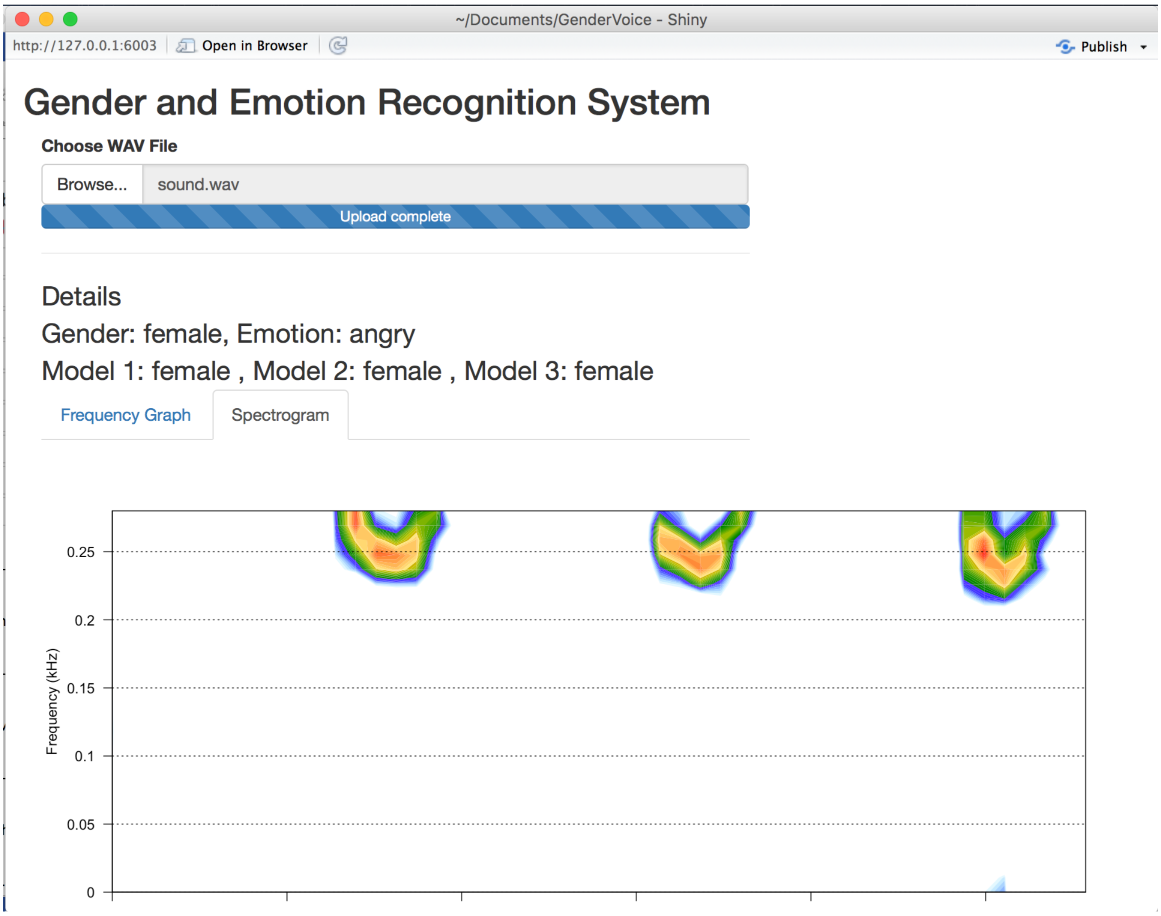


*FIG 5.8 Shiny App Output 2*

# CHAPTER 6

# CONCLUSIONS

The system provides facility to determine a person's gender and emotion from their voice sample provided in '.wav' format. The system predicts the gender and emotion accurately for most cases.

The system is provided with 3000+ voice samples divided as male, female for gender recognition model building. It has an accuracy of 97%.

Over 1000 voice samples are provided for emotion recognition model building. It predicts between 4 emotions which are neutral, angry, sad, fear.

Various models have shown various accuracy levels—

1. Random Forest- 0.86 or 86 %
2. SVM Model- 0.85 or 85 %
3. CART Model- 0.76 or 76%

# CHAPTER 7

# FUTURE WORK

This project is trained on voice samples to predict gender and emotion of a person. This program uses 3000 voice samples and three models for analysis which are Random Forest, SVM Model and CART Model. In future we can train the program on other models also like XGBoosted, GLM etc. This program predicts between four emotions - Anger, Neutral, Sad and Fear. In future we can train the model on more emotions like Joy, Grief etc.

# REFERENCES

The following links were very helpful during the completion of project:


1. https://cran.r-project.org/web/packages/seewave/index.html
2. https://cran.r-project.org/web/packages/tuneR/index.html
3. https://en.wikipedia.org/wiki/RStudio
4. https://shiny.rstudio.com/
5. https://tspace.library.utoronto.ca/handle/1807/24501
6. https://www.tutorialspoint.com/r/r_random_forest.html

# APPENDIX

## 1. R Files

*1.1. ui.R*

```
library(shiny)

fluidPage(
 titlePanel("Gender and Emotion Recognition System"),
   mainPanel(
     fileInput('file1', 'Choose WAV File', accept = c('audio/wav'), width = '100%'),
     tags$hr(),
     div(id='result', style='font-size: 22px;', htmlOutput('content')),

     conditionalPanel(condition='output.content != null',
                  tabsetPanel(id='graphs',
                               tabPanel('Frequency Graph', plotOutput("graph1", width=900,
height=450)),
                                    tabPanel('Spectrogram', plotOutput("graph2", width=900,
height=450))
                         ),

                  div(style='margin: 20px 0 0 0;')
     )
   )
 )
```

*1.2. server.R*

```
library(shiny)
library(tuneR)
library(seewave)
library(caTools)
library(rpart)
library(rpart.plot)
library(randomForest)
library(warbleR)
library(xgboost)
library(e1071)

source('analysis.R')

function(input, output) {

  v <- reactiveValues(data = NULL)
  observeEvent(input$file1, {
    content <- ''
    inFile <- input$file1

    if (grepl('.wav', tolower(inFile$name)) != TRUE) {
        content <- '<div class="shiny-output-error-validation">Please select a .WAV file to 
upload.</div>'
    }
    else if (!is.null(inFile)) {

      withProgress(message='Please wait ..', style='old', value=0, {
        inFile <- input$file1

        if (is.null(inFile))
          return(NULL)

        id <- sample(1:100000, 1)
        filePath <- paste0('./temp', sample(1:100000, 1), '/temp', id, '.wav')


        currentPath <- getwd()
        fileName <- basename(filePath)
        path <- dirname(filePath)
        dir.create(path)
        file.copy(inFile$datapath, filePath)
        content <- process(filePath)

        if (!is.null(content$graph1)) {
          output$graph1 <- content$graph1
```

```
        output$graph2 <- content$graph2
      }
      unlink(path, recursive = T)

    })
  }
  v$data <- formatResult(content)

})

output$content <- eventReactive(v$data, {
  HTML(v$data)
})


process <- function(path) {
  content1 <- list(label = 'Sorry, an error occurred.', prob = 0, data = NULL)
  graph1 <- NULL
  graph2 <- NULL
  freq <- list(minf = NULL, meanf = NULL, maxf = NULL)


    incProgress(0.3, message = 'Processing voice ..')
    content1 <- gender(path)
    incProgress(0.8, message = 'Building graph 1/2 ..')

    wl <- 2048
    ylim <- 280
    thresh <- 5

        freqs <- fund(content1$wave, fmax=ylim, ylim=c(0, ylim/1000), threshold=thresh,
plot=F, wl=wl)
    freq$minf <- round(min(freqs[,2], na.rm = T)*1000, 0)
    freq$meanf <- round(mean(freqs[,2], na.rm = T)*1000, 0)
    freq$maxf <- round(max(freqs[,2], na.rm = T)*1000, 0)

    graph1 <- renderPlot({
        fund(content1$wave, fmax=ylim, ylim=c(0, ylim/1000), type='l', threshold=thresh,
col='red', wl=wl)
      x <- freqs[,1]
      y <- freqs[,2] + 0.01
      labels <- freqs[,2]

      subx <- x[seq(1, length(x), 4)]
      suby <- y[seq(1, length(y), 4)]
      sublabels <- paste(round(labels[seq(1, length(labels), 4)] * 1000, 0), 'hz')
      text(subx, suby, labels = sublabels)
```

```
legend(0.5, 0.05, legend=c(paste('Min frequency', freq$minf, 'hz'), paste('Average frequency', freq$meanf, 'hz'), paste('Max frequency', freq$maxf, 'hz')), text.col=c('black', 'darkgreen', 'black'), pch=c(19, 19, 19))

    })

    incProgress(0.9, message = 'Building graph 2/2 ..')

    graph2 <- renderPlot({
      spectro(content1$wave, ovlp=40, zp=8, scale=FALSE, flim=c(0,ylim/1000), wl=wl)
    })


list(label=content1$label,cart=content1$cart,svm=content1$svm,emotion=content1$emotion, graph1=graph1, graph2=graph2)

  }

  formatResult <- function(result) {
    html<-""
    html <- paste0(html, '<div class="detail-header">Details</div>')
    html <- paste0(html, 'Gender: ', result$label, '<i class="fa fa-info" aria-hidden="true" title="Gender"></i>,  ')
    html <- paste0(html, 'Emotion: ',result$emotion, '<i class="fa fa-info" aria-hidden="true" title="Emotion"></i> <br> ')
    html <- paste0(html, 'Model 1: ',result$cart, '<i class="fa fa-info" aria-hidden="true" title="Model1"></i> , ')
    html <- paste0(html, 'Model 2: ',result$label, '<i class="fa fa-info" aria-hidden="true" title="Model2"></i> , ')
    html <- paste0(html, 'Model 3: ',result$svm, '<i class="fa fa-info" aria-hidden="true" title="Model3"></i>  ')
    html <- paste0(html, '</div>')
    html
  }


}
```

*1.3. analysis.R*

```
#packages <- c('tuneR', 'seewave', 'fftw', 'caTools', 'randomForest', 'warbleR', 'mice',
'e1071', 'rpart', 'rpart-plot', 'xgboost', 'e1071')
#if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
#  install.packages(setdiff(packages, rownames(installed.packages())))
#}

library(tuneR)
library(seewave)
library(caTools)
library(rpart)
library(rpart.plot)
library(randomForest)
library(warbleR)
library(xgboost)
library(e1071)

specan3 <- function(filepath){

 bp<- c(0,22)
 wl<- 2048
 threshold<- 5
 start<-0
 end<-20
 currentPath <- getwd()
 fileName <- basename(filepath)
 path <- dirname(filepath)

 print(path)
 print(fileName)

 setwd(path)
 print(getwd())
 r <- tuneR::readWave(fileName ,from = start, to = end, units = "seconds")


 b<- bp
 if(b[2] > ceiling(r@samp.rate/2000) - 1) b[2] <- ceiling(r@samp.rate/2000) - 1

 songspec <- seewave::spec(r, f = r@samp.rate, plot = FALSE)
 analysis <- seewave::specprop(songspec, f = r@samp.rate, flim = c(0, 280/1000), plot =
FALSE)

 meanfreq <- analysis$mean/1000
 sd <- analysis$sd/1000
 median <- analysis$median/1000
```

```
 Q25 <- analysis$Q25/1000
 Q75 <- analysis$Q75/1000
 IQR <- analysis$IQR/1000
 skew <- analysis$skewness
 kurt <- analysis$kurtosis
 sp.ent <- analysis$sh
 sfm <- analysis$sfm
 mode <- analysis$mode/1000
 centroid <- analysis$cent/1000

 peakf <- 0
 ff <- seewave::fund(r, f = r@samp.rate, ovlp = 50, threshold = threshold,
                 fmax = 280, ylim=c(0, 280/1000), plot = FALSE, wl = wl)[, 2]
 meanfun<-mean(ff, na.rm = T)
 minfun<-min(ff, na.rm = T)
 maxfun<-max(ff, na.rm = T)

 y <- seewave::dfreq(r, f = r@samp.rate, wl = wl, ylim=c(0, 280/1000), ovlp = 0, plot = F,
threshold = threshold, bandpass = b * 1000, fftw = TRUE)[, 2]
 meandom <- mean(y, na.rm = TRUE)
 mindom <- min(y, na.rm = TRUE)
 maxdom <- max(y, na.rm = TRUE)
 dfrange <- (maxdom - mindom)
 duration <- (end - start)

 changes <- vector()
 for(j in which(!is.na(y))){
  change <- abs(y[j] - y[j + 1])
  changes <- append(changes, change)
 }
if(mindom==maxdom) modindx<-0 else modindx <- mean(changes, na.rm = T)/dfrange

label<--1

x<-c(duration,meanfreq, sd, median, Q25, Q75, IQR, skew, kurt, sp.ent, sfm, mode,
        centroid, peakf, meanfun, minfun, maxfun, meandom, mindom, maxdom, dfrange,
modindx,label)

wave <<- r

names(x) <- c( "duration","meanfreq", "sd", "median", "Q25", "Q75", "IQR", "skew",
"kurt", "sp.ent",
                    "sfm","mode", "centroid", "peakf", "meanfun", "minfun", "maxfun",
"meandom", "mindom", "maxdom", "dfrange", "modindx","label")

return(list(acoustics = x, wave = wave))
}
```

```
gender <- function(filepath) {

result <- specan3(filepath)

test1 <- result$acoustics
test1$duration <- NULL
test1$sound.files <- NULL
test1$selec <- NULL
test1$peakf <- NULL
test1 <- test1[complete.cases(test1)]

wave <- result$wave

emotion <- getEmotion(test1)

setwd("/Users/prekshasingla/Documents/GenderVoice/")

data = read.csv("voice.csv")

set.seed(777)
spl <- sample.split(data$label, 0.7)
train <- subset(data, spl == TRUE)
test <- subset(data, spl == FALSE)


genderCART <- rpart(label ~ ., data=train, method='class')
prp(genderCART)
predictCART <- predict(genderCART,test1)[,2]

set.seed(777)
genderForest <- randomForest(label ~ ., data=train)
predictForest <- predict(genderForest, newdata=test1,type = "prob")[,2]

set.seed(777)
genderSvm <- svm(label ~ ., data=train, gamma=0.21, cost=8, probability=TRUE)
newdata <- data.frame(test1)
predictSvm <- predict(genderSvm, newdata, probability=TRUE)

if(predictCART > 0.5){
  result1<-"male"
}
if(predictCART <= 0.5){
  result1<-"female"
}
```

```
if(predictForest > 0.5){
  result2<-"male"
  test1$label<-"male"
}
if(predictForest <= 0.5){
  result2<-"female"
  test1$label<-"female"
}

list(label = test1$label,cart=result1,svm=predictSvm,emotion= emotion, wave = wave)

}

getEmotion <- function(test1){

  setwd("/Users/prekshasingla/Documents/GenderVoice/")

  data = read.csv("emotion.csv")

  set.seed(772)
  spl <- sample.split(data$label, 0.7)
  train <- subset(data, spl == TRUE)
  test <- subset(data, spl == FALSE)

  genderCART <- rpart(label ~ ., data=train, method='class')
  prp(genderCART)
  predictCART <- predict(genderCART,test1)[,2]

  set.seed(772)
  genderForest <- randomForest(label ~ ., data=train)
  predictForest1 <- predict(genderForest, newdata=test1)

  set.seed(772)
  genderSvm <- svm(label ~ ., data=train, gamma=0.21, cost=8, probability=TRUE)
  newdata <- data.frame(test1)
  predictSvm <- predict(genderSvm, newdata, probability=TRUE)

  return (predictForest1)
}
```

1.4. emotion_csv_create.R

```
#packages <- c('tuneR', 'seewave', 'fftw', 'caTools', 'randomForest', 'warbleR', 'mice',
'e1071', 'rpart', 'rpart-plot', 'xgboost', 'e1071')
#if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
#  install.packages(setdiff(packages, rownames(installed.packages())))
#}
library(tuneR)
library(seewave)
library(caTools)
library(rpart)
library(rpart.plot)
library(randomForest)
library(warbleR)
library(mice)
library(xgboost)
library(e1071)

specan3 <- function(X, bp = c(0,22), wl = 2048, threshold = 5){

   if(class(X) == "data.frame") {if(all(c("sound.files", "selec",
                                          "start", "end") %in% colnames(X)))
  {
    start <- as.numeric(unlist(X$start))
    end <- as.numeric(unlist(X$end))
    sound.files <- as.character(unlist(X$sound.files))
    selec <- as.character(unlist(X$selec))
  } else stop(paste(paste(c("sound.files", "selec", "start", "end")[!(c("sound.files", "selec",
                                                   "start", "end") %in% colnames(X))],
collapse=", "), "column(s) not found in data frame"))
  } else  stop("X is not a data frame")

  if(any(is.na(c(end, start)))) stop("NAs found in start and/or end")

  if(all(class(end) != "numeric" & class(start) != "numeric")) stop("'end' and 'selec' must
be numeric")

  if(any(end - start<0)) stop(paste("The start is higher than the end in", length(which(end
- start<0)), "case(s)"))

   if(any(end - start>20)) stop(paste(length(which(end - start>20)), "selection(s) longer
than 20 sec"))
  options( show.error.messages = TRUE)

  if(!is.vector(bp)) stop("'bp' must be a numeric vector of length 2") else{
    if(!length(bp) == 2) stop("'bp' must be a numeric vector of length 2")}
```

```
  fs <- list.files(path = getwd(), pattern = ".wav$", ignore.case = TRUE)
  if(length(unique(sound.files[(sound.files %in% fs)])) != length(unique(sound.files)))
        cat(paste(length(unique(sound.files))-length(unique(sound.files[(sound.files %in%
fs)])),
          ".wav file(s) not found"))

  d <- which(sound.files %in% fs)
  if(length(d) == 0){
    stop("The .wav files are not in the working directory")
  }  else {
    start <- start[d]
    end <- end[d]
    selec <- selec[d]
    sound.files <- sound.files[d]
  }

  x <- as.data.frame(lapply(1:length(start), function(i) {
      r <- tuneR::readWave(file.path(getwd(), sound.files[i]), from = start[i], to = end[i],
units = "seconds")

    b<- bp
    if(b[2] > ceiling(r@samp.rate/2000) - 1) b[2] <- ceiling(r@samp.rate/2000) - 1


    songspec <- seewave::spec(r, f = r@samp.rate, plot = FALSE)
    analysis <- seewave::specprop(songspec, f = r@samp.rate, flim = c(0, 280/1000), plot
= FALSE)

    meanfreq <- analysis$mean/1000
    sd <- analysis$sd/1000
    median <- analysis$median/1000
    Q25 <- analysis$Q25/1000
    Q75 <- analysis$Q75/1000
    IQR <- analysis$IQR/1000
    skew <- analysis$skewness
    kurt <- analysis$kurtosis
    sp.ent <- analysis$sh
    sfm <- analysis$sfm
    mode <- analysis$mode/1000
    centroid <- analysis$cent/1000

    peakf <- 0
    ff <- seewave::fund(r, f = r@samp.rate, ovlp = 50, threshold = threshold,
                    fmax = 280, ylim=c(0, 280/1000), plot = FALSE, wl = wl)[, 2]
    meanfun<-mean(ff, na.rm = T)
    minfun<-min(ff, na.rm = T)
    maxfun<-max(ff, na.rm = T)
```

```
   y <- seewave::dfreq(r, f = r@samp.rate, wl = wl, ylim=c(0, 280/1000), ovlp = 0, plot
= F, threshold = threshold, bandpass = b * 1000, fftw = TRUE)[, 2]
    meandom <- mean(y, na.rm = TRUE)
    mindom <- min(y, na.rm = TRUE)
    maxdom <- max(y, na.rm = TRUE)
    dfrange <- (maxdom - mindom)
    duration <- (end[i] - start[i])

    changes <- vector()
    for(j in which(!is.na(y))){
      change <- abs(y[j] - y[j + 1])
      changes <- append(changes, change)
    }
        if(mindom==maxdom) modindx<-0 else modindx <- mean(changes, na.rm = T)/
dfrange

    return(c(duration, meanfreq, sd, median, Q25, Q75, IQR, skew, kurt, sp.ent, sfm, mode,
             centroid, peakf, meanfun, minfun, maxfun, meandom, mindom, maxdom, dfrange,
modindx))
  }))

   rownames(x) <- c("duration", "meanfreq", "sd", "median", "Q25", "Q75", "IQR",
"skew", "kurt", "sp.ent",
                   "sfm","mode", "centroid", "peakf", "meanfun", "minfun", "maxfun",
"meandom", "mindom", "maxdom", "dfrange", "modindx")
  x <- data.frame(sound.files, selec, as.data.frame(t(x)))
  colnames(x)[1:2] <- c("sound.files", "selec")
  rownames(x) <- c(1:nrow(x))

  return(x)
}

processFolder <- function(folderName) {

  data <- data.frame()

  list <- list.files(pattern = '\\.wav')
  list

  for (fileName in list) {
    #print('hi')
    row <- data.frame(fileName, 0, 0, 5)
    data <- rbind(data, row)
  }

  names(data) <- c('sound.files', 'selec', 'start', 'end')
```

```
  acoustics <- specan3(data)

  setwd('..')

  acoustics
}

setwd("/Users/prekshasingla/Documents/GenderVoice/angry/")
angry <- processFolder('angry')
setwd("/Users/prekshasingla/Documents/GenderVoice/neutral/")
neutral <- processFolder('neutral')
setwd("/Users/prekshasingla/Documents/GenderVoice/sad/")
sad <- processFolder('sad')
setwd("/Users/prekshasingla/Documents/GenderVoice/fear/")
fear <- processFolder('fear')


neutral$label <- 1
angry$label <- 2
sad$label <- 3
fear$label <- 4

data <- rbind(neutral, angry, sad, fear)
data$label <- factor(data$label, labels=c('neutral','angry','sad', 'fear'))

data$duration <- NULL
data$sound.files <- NULL
data$selec <- NULL
data$peakf <- NULL

data <- data[complete.cases(data),]

write.csv(data, file='emotion.csv', sep=',', row.names=F)

set.seed(777)
spl <- sample.split(data$label, 0.7)
train <- subset(data, spl == TRUE)
test <- subset(data, spl == FALSE)

genderCART <- rpart(label ~ ., data=train, method='class')
prp(genderCART)
genderForest <- randomForest(label ~ ., data=train)

predictCART <- predict(genderCART)
predictCART.prob <- predictCART[,2]
table(train$label, predictCART.prob >= 0.5)
```

```
predictCART2 <- predict(genderCART, newdata=test)
predictCART2.prob <- predictCART2[,2]

predictForest <- predict(genderForest, newdata=train)
table(train$label, predictForest)

predictForest <- predict(genderForest, newdata=test)
table(test$label, predictForest)

set.seed(777)
genderSvm <- svm(label ~ ., data=train, gamma=0.21, cost=8)

predictSvm <- predict(genderSvm, train)
table(predictSvm, train$label)

predictSvm <- predict(genderSvm, test)
table(predictSvm, test$label)
```

*1.5. voice_csv_create.R*

```
#packages <- c('tuneR', 'seewave', 'fftw', 'caTools', 'randomForest', 'warbleR', 'mice',
'e1071', 'rpart', 'rpart-plot', 'xgboost', 'e1071')
#if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
#  install.packages(setdiff(packages, rownames(installed.packages())))
#}
library(tuneR)
library(seewave)
library(caTools)
library(rpart)
library(rpart.plot)
library(randomForest)
library(warbleR)
library(mice)
library(xgboost)
library(e1071)

specan3 <- function(X, bp = c(0,22), wl = 2048, threshold = 5){

  if(class(X) == "data.frame") {if(all(c("sound.files", "selec",
                                        "start", "end") %in% colnames(X)))
  {
    start <- as.numeric(unlist(X$start))
    end <- as.numeric(unlist(X$end))
    sound.files <- as.character(unlist(X$sound.files))
    selec <- as.character(unlist(X$selec))
  } else stop(paste(paste(c("sound.files", "selec", "start", "end")[!(c("sound.files", "selec",
                                                      "start", "end") %in% colnames(X))],
collapse=", "), "column(s) not found in data frame"))
  } else  stop("X is not a data frame")

  if(any(is.na(c(end, start)))) stop("NAs found in start and/or end")

  if(all(class(end) != "numeric" & class(start) != "numeric")) stop("'end' and 'selec' must
be numeric")

  if(any(end - start<0)) stop(paste("The start is higher than the end in", length(which(end -
start<0)), "case(s)"))

  if(any(end - start>20)) stop(paste(length(which(end - start>20)), "selection(s) longer
than 20 sec"))
  options( show.error.messages = TRUE)

  if(!is.vector(bp)) stop("'bp' must be a numeric vector of length 2") else{
    if(!length(bp) == 2) stop("'bp' must be a numeric vector of length 2")}
```

```
  fs <- list.files(path = getwd(), pattern = ".wav$", ignore.case = TRUE)
  if(length(unique(sound.files[(sound.files %in% fs)])) != length(unique(sound.files)))
    cat(paste(length(unique(sound.files))-length(unique(sound.files[(sound.files %in%
fs)])),
              ".wav file(s) not found"))

  d <- which(sound.files %in% fs)
  if(length(d) == 0){
    stop("The .wav files are not in the working directory")
  }  else {
    start <- start[d]
    end <- end[d]
    selec <- selec[d]
    sound.files <- sound.files[d]
  }

  x <- as.data.frame(lapply(1:length(start), function(i) {
    r <- tuneR::readWave(file.path(getwd(), sound.files[i]), from = start[i], to = end[i],
units = "seconds")

    b<- bp
    if(b[2] > ceiling(r@samp.rate/2000) - 1) b[2] <- ceiling(r@samp.rate/2000) - 1

    songspec <- seewave::spec(r, f = r@samp.rate, plot = FALSE)
    analysis <- seewave::specprop(songspec, f = r@samp.rate, flim = c(0, 280/1000), plot
= FALSE)

    meanfreq <- analysis$mean/1000
    sd <- analysis$sd/1000
    median <- analysis$median/1000
    Q25 <- analysis$Q25/1000
    Q75 <- analysis$Q75/1000
    IQR <- analysis$IQR/1000
    skew <- analysis$skewness
    kurt <- analysis$kurtosis
    sp.ent <- analysis$sh
    sfm <- analysis$sfm
    mode <- analysis$mode/1000
    centroid <- analysis$cent/1000

    peakf <- 0
    ff <- seewave::fund(r, f = r@samp.rate, ovlp = 50, threshold = threshold,
                    fmax = 280, ylim=c(0, 280/1000), plot = FALSE, wl = wl)[, 2]
    meanfun<-mean(ff, na.rm = T)
    minfun<-min(ff, na.rm = T)
    maxfun<-max(ff, na.rm = T)
```

```
    y <- seewave::dfreq(r, f = r@samp.rate, wl = wl, ylim=c(0, 280/1000), ovlp = 0, plot =
F, threshold = threshold, bandpass = b * 1000, fftw = TRUE)[, 2]
    meandom <- mean(y, na.rm = TRUE)
    mindom <- min(y, na.rm = TRUE)
    maxdom <- max(y, na.rm = TRUE)
    dfrange <- (maxdom - mindom)
    duration <- (end[i] - start[i])

    changes <- vector()
    for(j in which(!is.na(y))){
      change <- abs(y[j] - y[j + 1])
      changes <- append(changes, change)
    }
    if(mindom==maxdom) modindx<-0 else modindx <- mean(changes, na.rm = T)/
dfrange

    return(c(duration, meanfreq, sd, median, Q25, Q75, IQR, skew, kurt, sp.ent, sfm, mode,
             centroid, peakf, meanfun, minfun, maxfun, meandom, mindom, maxdom,
dfrange, modindx))
  }))

  rownames(x) <- c("duration", "meanfreq", "sd", "median", "Q25", "Q75", "IQR",
"skew", "kurt", "sp.ent",
                   "sfm","mode", "centroid", "peakf", "meanfun", "minfun", "maxfun",
"meandom", "mindom", "maxdom", "dfrange", "modindx")
  x <- data.frame(sound.files, selec, as.data.frame(t(x)))
  colnames(x)[1:2] <- c("sound.files", "selec")
  rownames(x) <- c(1:nrow(x))

  return(x)
}

processFolder <- function(folderName) {
  data <- data.frame()

  list <- list.files(pattern = '\\.wav')

  for (fileName in list) {
    row <- data.frame(fileName, 0, 0, 20)
    data <- rbind(data, row)
  }

  names(data) <- c('sound.files', 'selec', 'start', 'end')

  acoustics <- specan3(data)
  setwd('..')
  acoustics
```

```
}

gender <- function(filePath) {
  currentPath <- getwd()
  fileName <- basename(filePath)
  print(filePath)
  path <- dirname(filePath)

  setwd(path)
  data <- data.frame(fileName, 0, 0, 20)

  names(data) <- c('sound.files', 'selec', 'start', 'end')

  acoustics <- specan3(data)
  setwd(currentPath)
}


males <- processFolder('male')
females <- processFolder('female')

males$label <- 1
females$label <- 2
data <- rbind(males, females)
data$label <- factor(data$label, labels=c('male', 'female'))

data$duration <- NULL
data$sound.files <- NULL
data$selec <- NULL
data$peakf <- NULL

data <- data[complete.cases(data)]

write.csv(data, file='voice1.csv', sep=',', row.names=F)

set.seed(777)
spl <- sample.split(data$label, 0.7)
train <- subset(data, spl == TRUE)
test <- subset(data, spl == FALSE)

genderCART <- rpart(label ~ ., data=train, method='class')
prp(genderCART)
genderForest <- randomForest(label ~ ., data=train)

predictCART <- predict(genderCART)
predictCART.prob <- predictCART[,2]
table(train$label, predictCART.prob >= 0.5)
```

```
predictCART2 <- predict(genderCART, newdata=test)
predictCART2.prob <- predictCART2[,2]

predictForest <- predict(genderForest, newdata=train)
table(train$label, predictForest)

predictForest <- predict(genderForest, newdata=test)
table(test$label, predictForest)

set.seed(777)
genderSvm <- svm(label ~ ., data=train, gamma=0.21, cost=8)

predictSvm <- predict(genderSvm, train)
table(predictSvm, train$label)

predictSvm <- predict(genderSvm, test)
table(predictSvm, test$label)
```