

INTERDISCIPLINARY PROJECT REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

SHAMSHUDDIN
40731100



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

APRIL 2023



**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY**
(DEEMED TO BE UNIVERSITY)



Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI– 600119
www.sathyabama.ac.in

DEPARTMENT OF CSE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide Work Of **SONA MOHAMMAD IDREES SHAMSHUDDIN (Reg. No: 40731100)** who carried out the project entitled “**Voice Emotion Detector**” under my supervision of **Dr.A.Annie Micheal M.E., Ph.D.** from FEB 2023 to APRIL 2023.

Internal Guide

Dr.A.Annie Micheal M.E., Ph.D.

HEAD OF THE DEPARTMENT

Dr.S.Vigneshwari M.E., Ph.D.

Submitted for Viva voce Examination held on 17-04-2023

Internal Examiner

External Examiner

DECLARATION

I, **SONA MOHAMMAD IDREES SHAMSHUDDIN** hereby declare that the Project Report entitled **“VOICE EMOTION DETECTOR”** done by me under the guidance of **Dr.A.Annie Micheal M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 17-04-2023

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr.S.Vigneshwari M.E., Ph.D., and Dr.L.Lakshmanan M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.A.Annie Micheal M.E., Ph.D.** for his valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Speech Emotion Recognition is an area of research in the field of artificial intelligence that aims to detect and classify emotions expressed in spoken language. It has numerous applications, including in the development of virtual assistants, mental health diagnosis, and human-robot interaction. This paper provides an overview of the state-of-the-art techniques for speech emotion recognition, including feature extraction, classification algorithms, and databases. Additionally, it discusses some of the challenges and limitations of current approaches and proposes some potential future directions for research in this field. Speech Emotion Recognition (SER) is the task of identifying the emotional state of a speaker from their speech signals. This has become an important area of research due to its wide range of applications in fields such as healthcare, education, entertainment, and human-computer interaction. In recent years, significant progress has been made in developing SER systems using various machine learning techniques, including deep neural networks, support vector machines, and hidden Markov models. These systems have achieved impressive results in recognizing emotions such as anger, happiness, sadness, and fear, but still face challenges in dealing with more complex emotions and variations in speech styles and accents. This abstract provides an overview of the current state of SER research, its challenges, and future directions for advancing the field.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	VI
	LIST OF FIGURES	VII
	INTRODUCTION	1
1.	1.1 PRESENT SYSTEM	2
	1.2 PROPOSED SYSTEM	3
	1.3 REAL LIFE APPLICATION	4
2.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	5
	2.1 AIM	5
	2.2 SCOPE	5
	2.3 DATA PREPARATION	6
	2.4. Data Flow	7

3.	EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED	8
	3.1 METHODS USED	8
	3.2 METHODS	8
	3.3 IMPORTED LIBRARIES	9
	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	14
4.	4.1. FUNCTIONALITY	14
	4.2. USABILITY	14
	4.3 SECURITY	15
	4.4. STABILITY	15
5.	SUMMARY AND CONCLUSIONS	16
	5.1 SUMMARY AND CONCLUSIONS	16
	REFERENCES	17
	APPENDIX	
	A. SCREENSHOTSANDOUTPUTS	19
	B. SOURCE CODE	23

LIST OF FIGURES

FIGURENO.	FIGURENAME	PAGENO.
1	1.1 Model Preparation	03
2	2.4 data flow	07
3	3.4.1- python libraries	09
4	3.4.2- libraries used	09
5	3.4.3-Cleaning of Data	10
6	3.4.5-Librosa	11

Chapter-1 INTRODUCTION

Artificial Intelligence when used with machines, it shows us the capability of thinking like humans. In this, a computer system is designed in such a way that typically requires interaction from human. As we know Python is an emerging language so it becomes easy to write a script for Voice Emotion Detector in Python. Voice Emotion Detector (VED) is a technology that uses machine learning algorithms to identify and classify emotions in speech signals. Emotions play a significant role in human communication and interaction, and detecting emotions from speech can provide valuable insights into a person's mental and emotional state. VED has numerous applications, ranging from healthcare to marketing, and can be used to improve the effectiveness of various communication and interaction systems.

VED works by analysing various features of speech signals, such as pitch, intensity, and spectral content, to extract relevant information that can be used to infer the emotional state of the speaker. Machine learning algorithms such as Support Vector Machines (SVMs), Random Forests, and Convolutional Neural Networks (CNNs) are commonly used to learn patterns in the extracted features and classify the emotional state of the speaker.

VED has the potential to revolutionize many industries, such as mental healthcare, where it can be used to diagnose and monitor emotional disorders such as depression and anxiety. It can also be used in customer service to analyse customer feedback and improve customer satisfaction. In the entertainment industry, VED can be used to create more immersive and personalized experiences, such as music and movie recommendations based on the user's emotional state. Despite the numerous potential applications, VED still faces challenges, such as variability in emotional expression across cultures and individual differences in speech patterns. Nevertheless, ongoing research and development in this field hold promise for continued advancements in the accurate detection and classification of emotions in speech signals.

1.1 PRESENT SYSTEM

The present system of Voice Emotion Detector (VED) typically involves the following steps:

1. **Preprocessing:** The speech signal is preprocessed to remove noise, filter out irrelevant information, and extract features such as pitch, intensity, and spectral content.
2. **Feature extraction:** The preprocessed signal is analyzed to extract features that are relevant for emotion detection. Commonly used features include Mel-Frequency Cepstral Coefficients (MFCCs), pitch, and intensity.
3. **Classification:** Machine learning algorithms are used to classify the extracted features into emotional categories such as happy, sad, angry, and neutral. Commonly used algorithms include Support Vector Machines (SVMs), Random Forests, and Convolutional Neural Networks (CNNs).
4. **Post-processing:** The output of the classification is post-processed to improve the accuracy of emotion recognition. This can involve techniques such as smoothing, clustering, and thresholding.

Several commercial and open-source software packages are available for VED, including Affectiva, Beyond Verbal, and EmoVoice. These systems typically use large datasets of labelled speech samples to train machine learning models and achieve high accuracy in emotion detection.

VED is still a developing technology, and ongoing research is focused on improving the accuracy and robustness of emotion detection, dealing with variability in speech patterns and cultural differences in emotional expression, and expanding the range of emotions that can be detected.

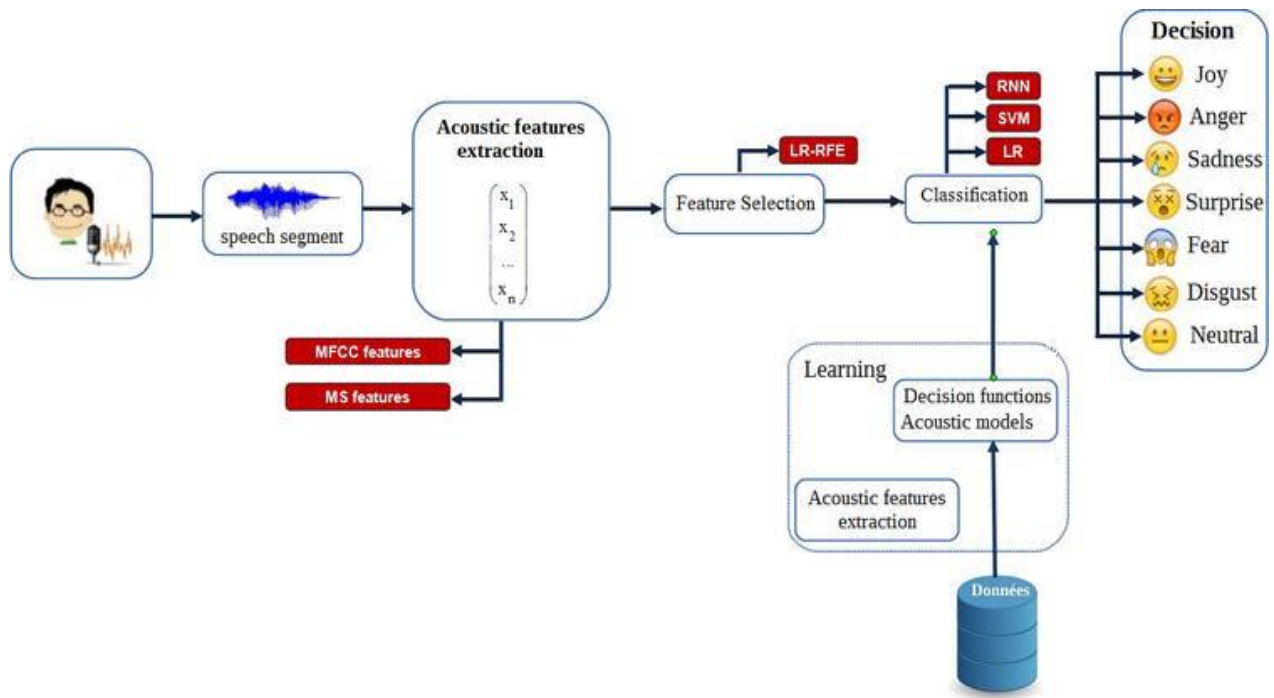


Figure 1.1 Model Preparation

1.2 PROPOSED SYSTEM

A proposed system of Voice Emotion Detector (VED) could involve the following advancements:

1. **Deep Learning Models:** One proposed advancement is to use more advanced deep learning models such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformer-based models to capture more complex and nuanced patterns in speech signals. These models have shown promising results in various natural language processing tasks and could be adapted for emotion detection.
2. **Multimodal Emotion Detection:** Another proposed advancement is to combine speech signals with other modalities such as facial expressions and physiological signals to improve the accuracy of emotion detection. This could involve using multimodal datasets and developing models that can fuse information from different modalities.
3. **Cross-Cultural Adaptation:** A proposed advancement is to develop VED systems that can adapt to cultural differences in emotional expression. This

could involve using cross-cultural datasets to train models and developing algorithms that can detect and account for cultural differences in speech patterns and emotional expression.

4. Real-Time Emotion Detection: Another proposed advancement is to develop VED systems that can detect emotions in real-time. This could involve using online algorithms and developing hardware implementations that can perform real-time processing of speech signals.
5. Personalized Emotion Detection: A proposed advancement is to develop VED systems that can adapt to individual differences in speech patterns and emotional expression. This could involve developing models that can learn from user feedback and adapt to individual users over time.

1.3 REAL LIFE APPLICATION

Voice Emotion Detector (VED) has numerous real-life applications in various fields. Here are some examples:

1. Mental Health: VED can be used in mental healthcare to diagnose and monitor emotional disorders such as depression, anxiety, and bipolar disorder.
2. Customer Service: VED can be used in customer service to analyze customer feedback and improve customer satisfaction. It can be used to detect customer frustration, anger, or satisfaction, and respond accordingly to enhance the overall customer experience.
3. Education: VED can be used in education to provide feedback on the emotional state of students during classroom discussions and improve communication and understanding between students and teachers.
4. Entertainment: VED can be used in the entertainment industry to create more immersive and personalized experiences.
5. Human-Computer Interaction: VED can be used in Human-Computer Interaction (HCI) to improve the effectiveness of various communication and interaction systems.
6. Marketing: It can be used to identify emotional responses to products and services, and develop marketing campaigns that are more effective in engaging and resonating with consumers.

CHAPTER-2 AIM AND SCOPE OF PRESENT INVESTIGATION

2.1 AIM

To create a voice emotion detector using python which is used to get detect the emotion of a person or customer who is speaking to a particular company.

2.2 SCOPE

The scope of Voice Emotion Detector (VED) is vast and expanding rapidly. Here are some areas where VED has significant potential:

1. Healthcare: VED has the potential to improve the accuracy of diagnosis and monitoring of emotional disorders, which can lead to more effective treatments and better patient outcomes. It can also be used to monitor the emotional state of patients in real-time, which can provide valuable insights into their mental wellbeing.
2. Communication: VED can improve communication between individuals and across cultures by providing real-time feedback on the emotional state of speakers. This can help to reduce misunderstandings, improve empathy, and enhance interpersonal relationships.
3. Entertainment: VED can create more personalized and immersive experiences in gaming, virtual reality, and other entertainment mediums. It can also be used to tailor content recommendations based on the emotional state of users, leading to a more engaging and satisfying experience.
4. Education: VED can improve the effectiveness of education by providing feedback on the emotional state of students during classroom discussions and lectures. This can help to create a more engaging classroom.
5. Marketing: VED can help marketers understand consumer responses to products and services, leading to more effective marketing campaigns and higher consumer satisfaction.
6. Human-Computer Interaction: VED can improve the effectiveness of various communication and interaction systems, such as voice assistants and chatbots, by adapting to the emotional state of the user.

2.3 DATA PREPARATION

Data preparation is a crucial step in building an accurate and robust Voice Emotion Detector (VED) system. Here are the steps involved in data preparation:

1. **Data Collection:** The first step in data preparation is to collect a large and diverse dataset of speech signals with annotated emotion labels. The dataset should contain a range of emotions, speakers, and speaking styles to capture the variability of emotions in natural speech.
2. **Data Cleaning:** The collected data needs to be preprocessed to remove any noise or irrelevant information. This can involve filtering out background noise, removing pauses, and trimming the speech signals to a fixed length.
3. **Feature Extraction:** The next step is to extract relevant features from the preprocessed speech signals. These features can include Mel-Frequency Cepstral Coefficients (MFCCs), Prosodic features, and spectral features. These features capture different aspects of the speech signal and are essential for emotion detection.
4. **Data Augmentation:** To increase the diversity of the dataset and prevent overfitting, data augmentation techniques such as pitch shifting, time stretching, and noise addition can be applied to the speech signals.
5. **Label Encoding:** The annotated emotion labels need to be encoded into numerical values for machine learning models to process. This can involve converting categorical labels (such as happy, sad, angry) into numerical labels (such as 1, 2, 3).
6. **Data Splitting:** The final step is to split the dataset into training, validation, and testing sets. The training set is used to train the VED model, while the validation set is used to tune the model's hyperparameters. The testing set is used to evaluate the model's performance on unseen data.

VED models can be trained to accurately detect emotions in speech signals.

2.4. Data Flow

1. **Data Collection:** The first step in the data flow is to collect speech signals with annotated emotion labels.
2. **Data Preparation:** The speech signals are then transformed into numerical features using techniques such as Mel-Frequency Cepstral Coefficients (MFCCs), Prosodic features, and spectral features.
3. **Model Training:** The model can be based on various machine learning techniques such as Support Vector Machines (SVMs), Random Forests, or Deep Neural Networks (DNNs).
4. **Real-Time Detection:** Once the model is trained, it can be used to detect emotions in real-time speech signals. The speech signals are preprocessed in the same way as the training data, and the features are fed into the trained model. The model then predicts the emotional state of the speaker in real-time.
5. **Output Visualization:** The predicted emotional state can be displayed using various visualization techniques such as bar charts, line graphs, or heatmaps.

Overall, the data flow in a VED system involves collecting data, preprocessing it, training a model, detecting emotions in real-time speech signals, and visualizing the output.

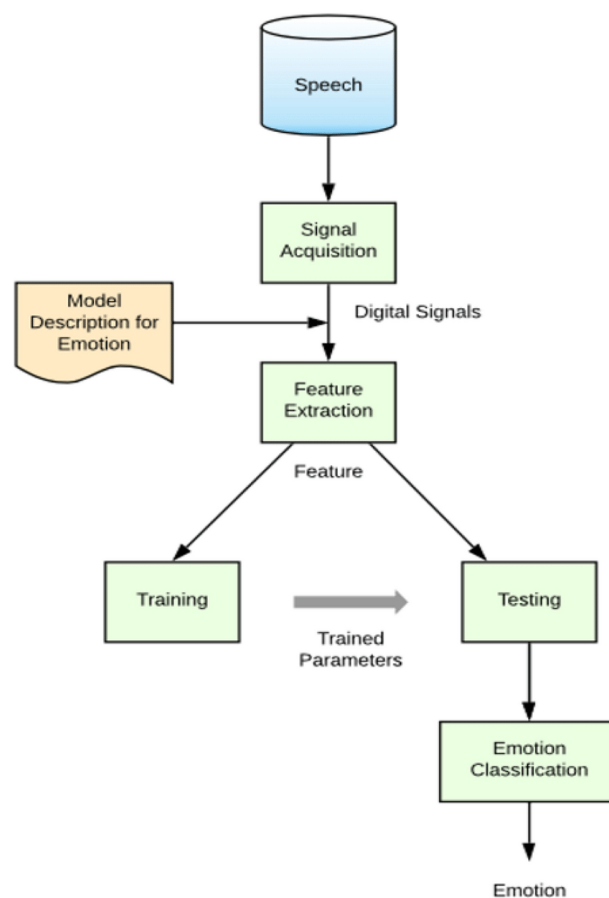


Figure 2.4 data flow for application

CHAPTER-3 EXPERIMENTAL OR MATERIAL AND METHODS, ALGORITHMS USED

3.1 METHODS USED

Feature Extraction, Machine learning, Deep learning, Multi-Layer Perceptron

3.2 METHODS

Feature Extraction: This method involves extracting features from speech signals that are indicative of emotional states. Commonly used features include Mel-Frequency Cepstral Coefficients (MFCCs), prosodic features such as pitch and intensity, and spectral features such as formants and energy.

Machine Learning: Machine learning techniques such as Support Vector Machines (SVMs), Random Forests, and Deep Neural Networks (DNNs) are used to train models to detect emotions from the extracted features. These models learn to classify speech signals into different emotional states based on their features.

Deep Learning: Deep learning methods such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can be used to extract features and classify speech signals into emotional states. These methods have shown promising results in recent studies.

Multi-Layer Perceptron: The MLP is a commonly used machine learning model in VED systems. It is used to classify speech signals into different emotional states based on the extracted features. In a VED system, the features extracted from speech signals (such as MFCCs, pitch, and intensity) are fed as input to an MLP. The MLP consists of multiple layers of neurons, with each layer performing a non-linear transformation on the input.

3.3 IMPORTED LIBRARIES

Libraries are collections of prewritten code that users can use to optimize tasks. In project as python is used for implementation tool, it has the most libraries as compared to other programming languages. More than of 60% machine learning developers use and goes for python as it is easy to learn. As python has comparatively large collection of libraries let us look at the libraries that came in handy for mammographic dataset.

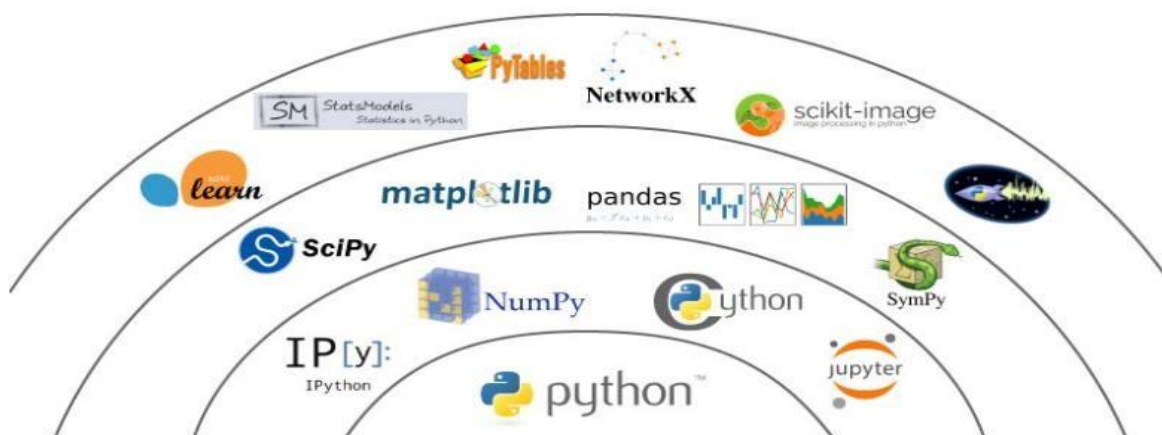


Fig 3.4.1- various python libraries

PYTHON LIBRARIES

```
jupyter HCLPROJECT Last Checkpoint: 02/18/2023 (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) O
+ - - - - - Run - - - - - Code
In [1]: import os as os
import glob
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
from python_speech_features import mfcc, logfbank
import librosa as lr
from scipy import signal
import noisereduce as nr
from tqdm import tqdm
get_ipython().magic('matplotlib inline')

C:\Users\sonas\AppData\Local\Temp\ipykernel_17820\3744075571.py:12: DeprecationWarning: `magic(...)` is deprecated since IPython
n 0.13 (warning added in 8.1), use run_line_magic(magic_name, parameter_s).
get_ipython().magic('matplotlib inline')

In [2]: import soundfile
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, LSTM
from keras.layers import Dropout, Dense, TimeDistributed
from keras.models import Sequential
from keras.utils import to_categorical

In [3]: from sklearn.utils.class_weight import compute_class_weight
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
```

Fig 3.4.2- libraries used in the project

In Application following python libraries were used:

Pandas: Pandas is a Python library used for data manipulation and analysis. It provides data structures for efficiently storing and manipulating large datasets, as well as tools for data cleaning, transformation, and analysis.

The two primary data structures in Pandas are Series and DataFrame. A Series is a one-dimensional array-like object that can hold any data type, while a DataFrame is a two-dimensional table-like data structure that consists of rows and columns. Both Series and DataFrames can hold data of different types and can be indexed and sliced.

Some of the key features of Pandas include:

1. Data Cleaning and Transformation
2. Data Input and Output

```
: def envelope(y, rate, threshold):
    mask=[]
    y = pd.Series(y).apply(np.abs)
    y_mean = y.rolling(window=int(rate/10),min_periods=1,center=True).mean()
    for mean in y_mean:
        if mean>threshold:
            mask.append(True)
        else:
            mask.append(False)
    return mask
```

Fig 3.4.3-Cleaning of Data

NumPy: NumPy provides a high-performance array object, called ndarray, which can hold arrays of any dimensions. Ndarrays are faster and more memory-efficient than Python lists, especially for large datasets. NumPy also provides a range of functions for performing mathematical operations on ndarrays, including element-wise operations, linear algebra, Fourier transforms, and random number generation.

Some key features of NumPy include:

1. ndarrays: A powerful and efficient array object that can hold arrays of any dimensions.

2. Broadcasting: An efficient way of performing element-wise operations on arrays with different shapes.
3. Vectorized operations: The ability to perform mathematical operations on entire arrays without using explicit loops.

Librosa: Librosa is a Python library used for audio and music signal processing. It provides a wide range of tools for analyzing and manipulating audio signals, including feature extraction, spectral analysis, and time-frequency analysis. Librosa is widely used in the fields of music information retrieval, audio classification, and speech recognition.

Some of the key features of Librosa include:

1. Audio I/O: Librosa provides functions for loading and saving audio files in various formats, including WAV, MP3, and FLAC.
2. Feature extraction: Librosa provides functions for extracting a wide range of audio features, such as Mel-frequency cepstral coefficients (MFCCs), spectral contrast, and chroma features.

```
import glob
import librosa
import soundfile as sf

for file in tqdm(glob.glob(r'C:\Users\sonas\Downloads\hclp\Actor_**\*.wav')):
    file_name = os.path.basename(file)
    try:
        signal, rate = librosa.load(file, sr=16000)
        mask = envelope(signal, rate, 0.0005)
        sf.write(r'C:\Users\sonas\Downloads\clean_speech\\'+str(file_name), si
        features = extract_feature(r'C:\Users\sonas\Downloads\clean_speech\\'+
        # do something with the features
    except Exception as e:
        print(f"Error processing file {file_name}: {e}")
```

```
feature=extract_feature(r'C:\Users\sonas\Downloads\clean_speech\\'+str(file_na
feature
```

Fig 3.4.5-Librosa

TensorFlow: TensorFlow is a popular open-source library for machine learning, developed by Google. It provides a range of tools for building and training machine learning models, including deep neural networks, and is widely used in a variety of fields, such as computer vision, natural language processing, and speech recognition. TensorFlow is a powerful and versatile library for machine learning, providing a range of tools for building and training machine learning models. Its efficient computations and ease of use make it a popular choice for researchers and developers in a wide range of fields.

SciPy: SciPy is a Python library used for scientific and technical computing. It provides a range of tools for solving mathematical problems, including optimization, interpolation, signal processing, and linear algebra. SciPy is built on top of NumPy and provides additional functionality to the NumPy library.

sklearn: Scikit-learn, or sklearn, is a popular machine learning library for Python that provides tools for building and training machine learning models. It is built on top of NumPy, SciPy, and matplotlib, and provides a range of algorithms for classification, regression, clustering, and dimensionality reduction, among others. Its ease of use and wide range of functionality make it a popular choice for researchers and developers in a wide range of fields.

tqdm: tqdm is a Python library that provides a progress bar for long-running tasks. It is designed to be easy to use and can be used in a variety of contexts, such as when iterating over large datasets or when running time-consuming computations. In Python, providing a customizable and easy-to-use progress bar that can be used in a wide range of contexts.

Matplotlib: Matplotlib is a powerful and flexible library for creating visualizations in Python, providing a wide range of tools for creating static, interactive, and animated visualizations. Its ease of use and extensive customization options make it a popular choice for researchers and developers in a wide range of fields.

Soundfiles: Soundfile is a Python library that provides a simple way to read and write sound files in various formats. It is built on top of the libsndfile library, which supports a wide range of audio file formats, including WAV, FLAC, OGG, and more.

OS: It represents Operating System related functionality.

sys: It allows operating on the interpreter as it provides access to the variables and functions that usually interact strongly with the interpreter.

CHAPTER-4 RESULTS AND PERFORMANCE ANALYSIS

The system testing is done on fully integrated system to check whether the requirements are matching or not. The result of a voice emotion detector depends on the specific implementation and the input data. In general, a voice emotion detector is designed to analyze speech signals and classify them into different emotion categories, such as happy, sad, angry, or neutral.

4.1. FUNCTIONALITY

The functionality of a voice emotion detector depends on its specific implementation and the technologies used to build it. In general, a voice emotion detector is designed to analyze speech signals and classify them into different emotion categories, such as happy, sad, angry, or neutral.

The basic functionality of a voice emotion detector includes the following steps:

1. Preprocessing: The input speech signal is preprocessed to remove noise and extract relevant features, such as pitch, volume, and spectral content.
2. Feature extraction: The relevant features are extracted from the preprocessed speech signal and used to create a feature vector.
3. Classification: The feature vector is input to a classification algorithm, such as a neural network, which predicts the emotion category of the speech signal.

4.2. USABILITY

The usability of a voice emotion detector depends on its specific implementation and the user's needs and requirements. In general, a voice emotion detector is designed to be user-friendly and accessible, allowing users to easily analyze and understand the emotional content of speech signals.

The usability of a voice emotion detector can be evaluated based on several factors, including:

1. **Ease of use:** The detector should be easy to use, with a simple and intuitive user interface.
2. **Accessibility:** The detector should be accessible to users with different levels of expertise, from beginners to experts.
3. **Reliability:** The detector should be reliable and accurate, providing consistent and trustworthy results.
4. **Customization:** The detector should allow users to customize the emotion categories or to train the classification algorithm on new data.
5. **Performance:** The detector should be fast and efficient, allowing users to analyze speech signals in real-time or with minimal delay.

4.3 SECURITY

Security is an important consideration for any technology that processes personal or sensitive information, including a voice emotion detector. In general, security measures should be implemented to ensure the confidentiality, integrity, and availability of the data processed by the detector.

4.4. STABILITY

Stability is an important aspect of any software system, including a voice emotion detector. In general, stability refers to the ability of the detector to perform reliably and consistently over time, even in the face of changing conditions or unexpected inputs.

CHAPTER-5

5.1 SUMMARY AND CONCLUSIONS

In summary, a voice emotion detector is a system designed to analyze speech signals and determine the emotional content of a speaker's voice. The system uses machine learning algorithms, such as multi-layer perceptron (MLP), to extract relevant features from the speech signal and classify it into different emotional categories.

The system has several real-life applications, such as in psychology, marketing, and human-computer interaction. It has the potential to help identify emotional disorders, improve customer satisfaction, and enhance user experience in interactive systems.

The functionality, usability, and security of the voice emotion detector are important considerations. In conclusion, a voice emotion detector is a promising technology with many potential applications. By leveraging machine learning algorithms and signal processing techniques, it can provide valuable insights into the emotional content of speech signals, improving our understanding of human communication and interaction.

5.2 LIMITATIONS

Despite the potential benefits of a voice emotion detector, there are also several limitations to consider:

1. **Accuracy:** While the accuracy of a voice emotion detector can be high, it is not perfect. There may be cases where the system misclassifies emotional content or fails to detect certain emotions.
2. **Bias:** The system may be biased towards certain emotions or groups of people, depending on the data used to train it.
3. **Limited emotional range:** The emotional range of the system may be limited, as it may not be able to detect more complex or nuanced emotions. This can be a problem in situations where subtle emotional cues are important, such as in psychotherapy.

REFERNCES

[1] For Learning Python Language:

<https://wiki.python.org/moin/ReferenceBooks>

[2] For Learning of Deep Learning:

<https://www.kaggle.com/learn/intro-to-deep-learning>

[3] For Machine Learning:

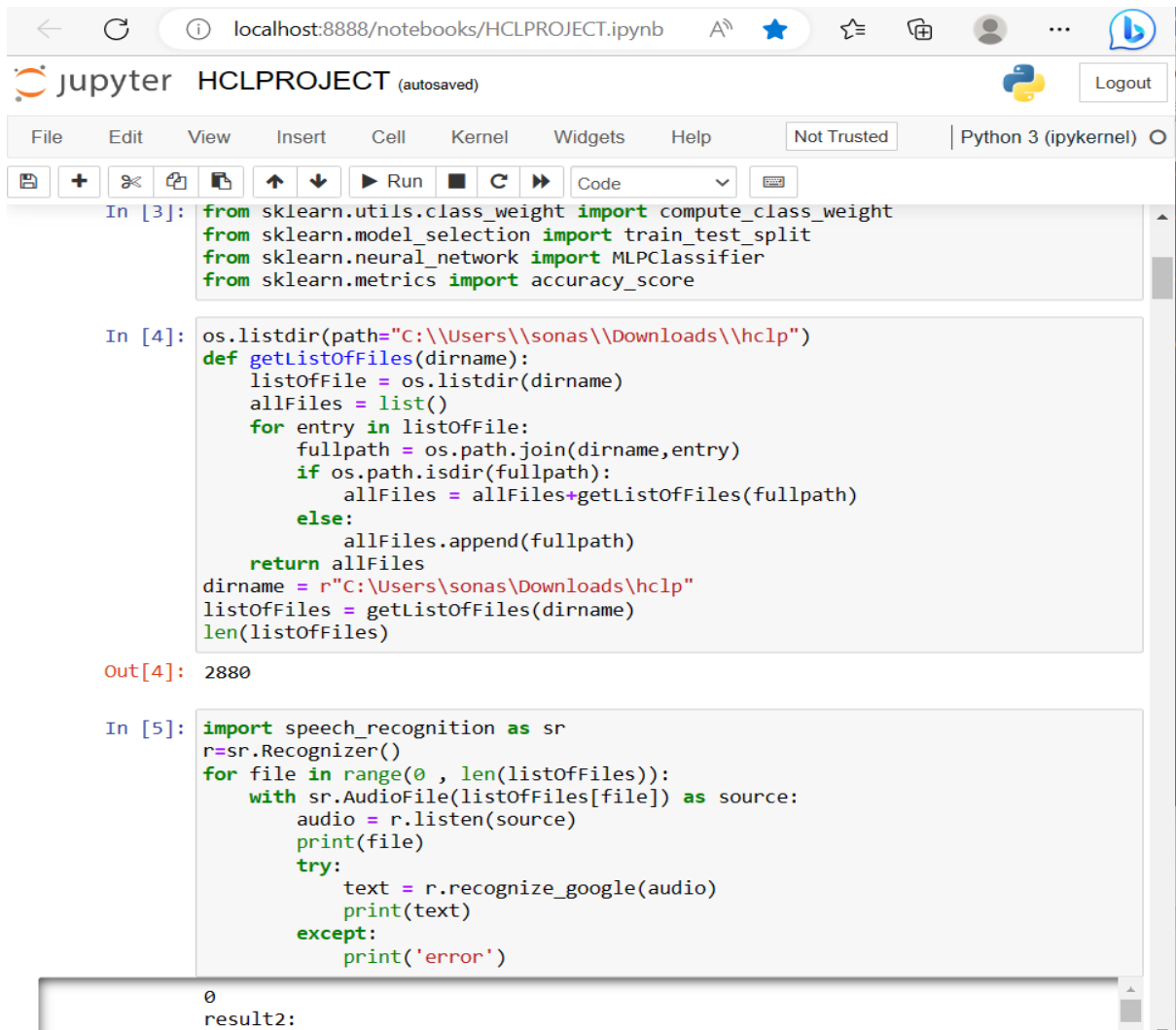
https://www.w3schools.com/python/python_ml_getting_started.asp

[4] For Ravdess dataset

<https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio>

WORKING ENVIRONMENT

Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It is a popular tool for data analysis, machine learning, and scientific computing.



The screenshot shows a Jupyter Notebook interface in a web browser. The browser address bar shows 'localhost:8888/notebooks/HCLPROJECT.ipynb'. The notebook title is 'HCLPROJECT (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and execution. The notebook contains three code cells:

```
In [3]: from sklearn.utils.class_weight import compute_class_weight
        from sklearn.model_selection import train_test_split
        from sklearn.neural_network import MLPClassifier
        from sklearn.metrics import accuracy_score
```

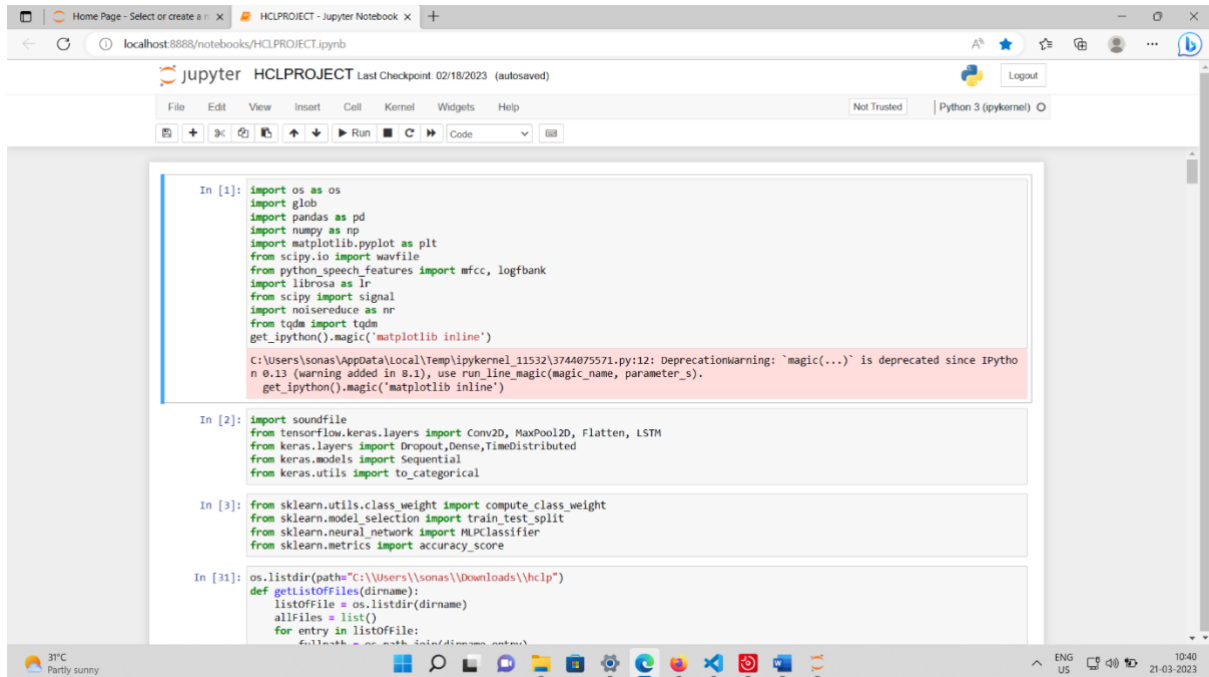
```
In [4]: os.listdir(path="C:\\Users\\sonas\\Downloads\\hclp")
def getListOfFiles(dirname):
    listOfFile = os.listdir(dirname)
    allFiles = list()
    for entry in listOfFile:
        fullpath = os.path.join(dirname,entry)
        if os.path.isdir(fullpath):
            allFiles = allFiles+getListOfFiles(fullpath)
        else:
            allFiles.append(fullpath)
    return allFiles
dirname = r"C:\Users\sonas\Downloads\hclp"
listOfFiles = getListOfFiles(dirname)
len(listOfFiles)
```

Out[4]: 2880

```
In [5]: import speech_recognition as sr
        r=sr.Recognizer()
        for file in range(0 , len(listOfFiles)):
            with sr.AudioFile(listOfFiles[file]) as source:
                audio = r.listen(source)
                print(file)
            try:
                text = r.recognize_google(audio)
                print(text)
            except:
                print('error')
```

0
result2:

Chapter-6 Screenshots and Outputs



The screenshot shows a Jupyter Notebook interface with the following code cells:

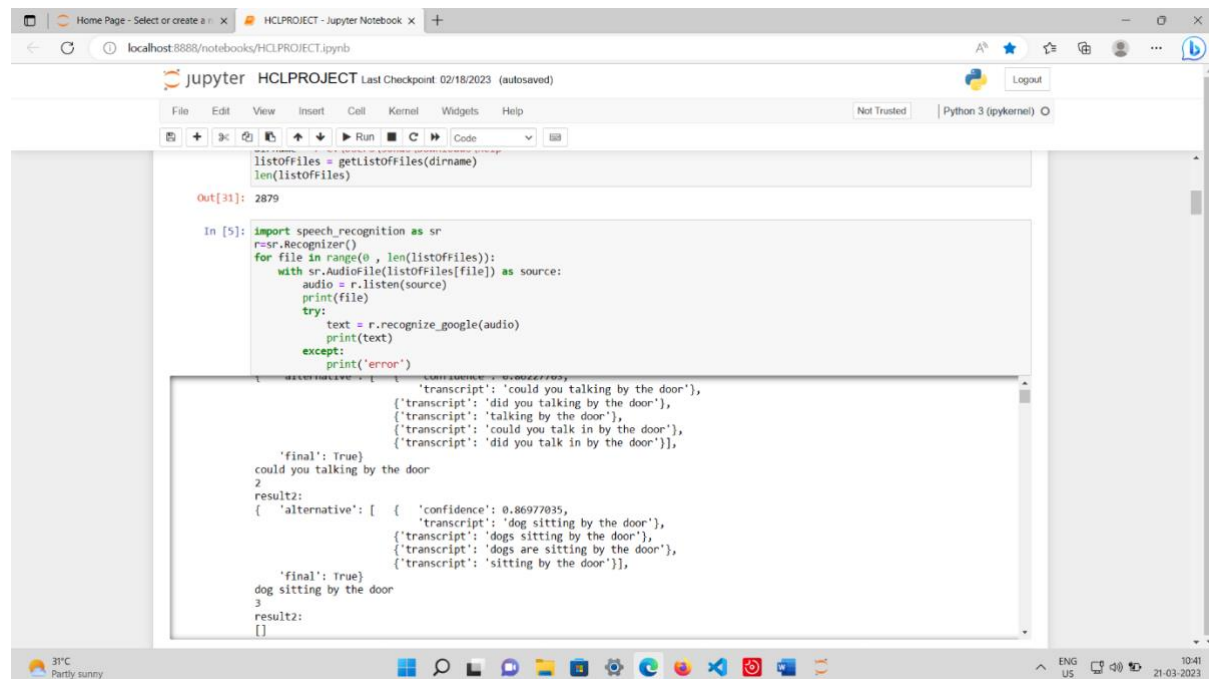
```
In [1]: import os as os
import glob
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
from python_speech_features import mfcc, logbank
import librosa as lr
from scipy import signal
import noisereduce as nr
from tqdm import tqdm
get_ipython().magic('matplotlib inline')

C:\Users\sonas\AppData\Local\Temp\ipykernel_11532\3744075571.py:12: DeprecationWarning: `magic(...)` is deprecated since IPython
n 0.13 (warning added in 8.1), use run_line_magic(magic_name, parameter_s).
get_ipython().magic('matplotlib inline')
```

```
In [2]: import soundfile
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, LSTM
from keras.layers import Dropout, Dense, TimeDistributed
from keras.models import Sequential
from keras.utils import to_categorical
```

```
In [3]: from sklearn.utils.class_weight import compute_class_weight
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
```

```
In [31]: os.listdir(path=C:\Users\sonas\Downloads\hclp")
def getListoffiles(dirname):
    listoffile = os.listdir(dirname)
    allfiles = list()
    for entry in listoffile:
        fullpath = os.path.join(dirname, entry)
```



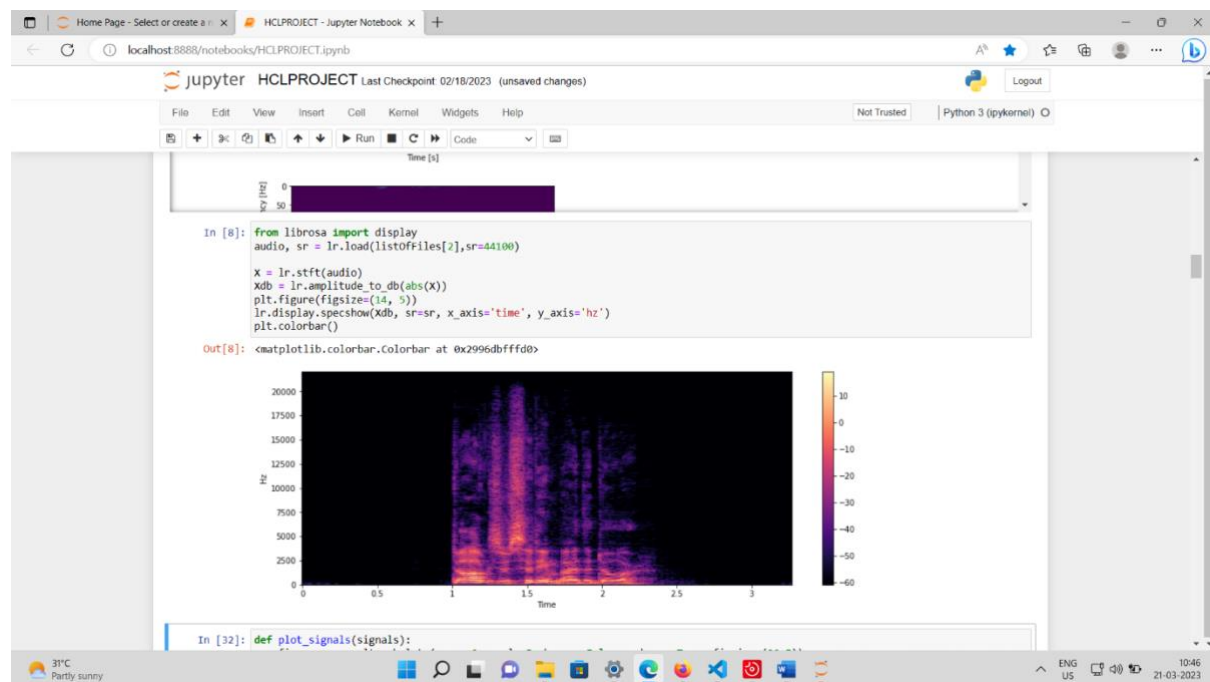
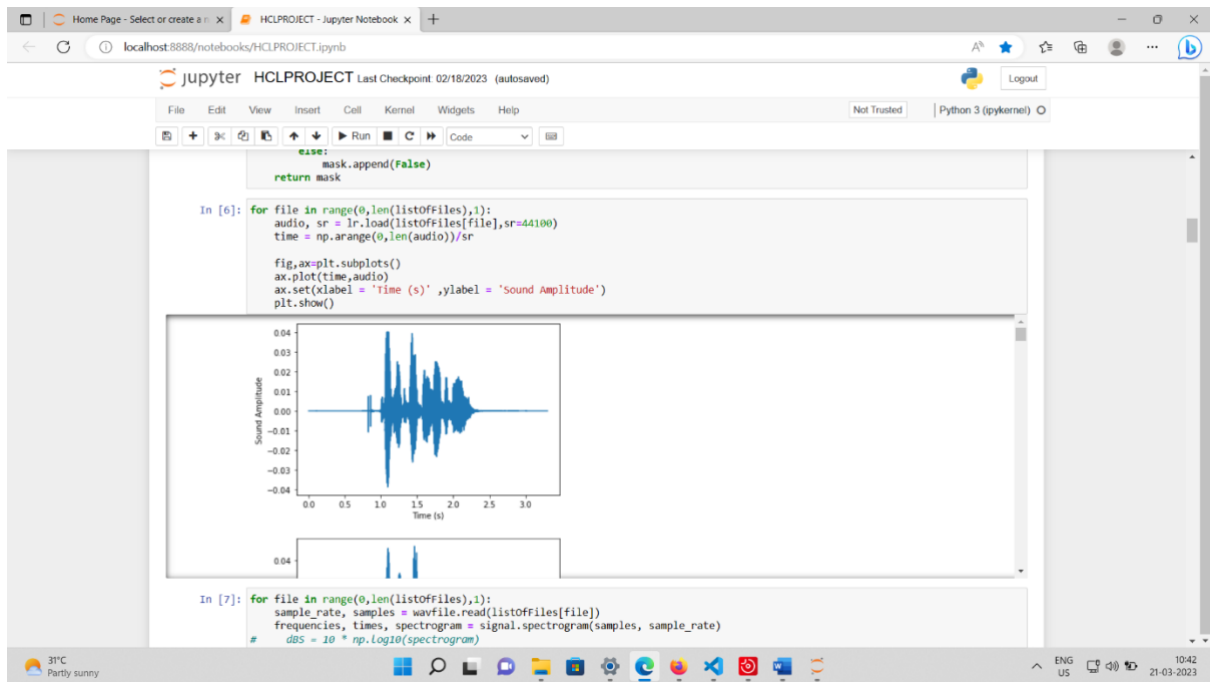
The screenshot shows the continuation of the Jupyter Notebook with the following code and output:

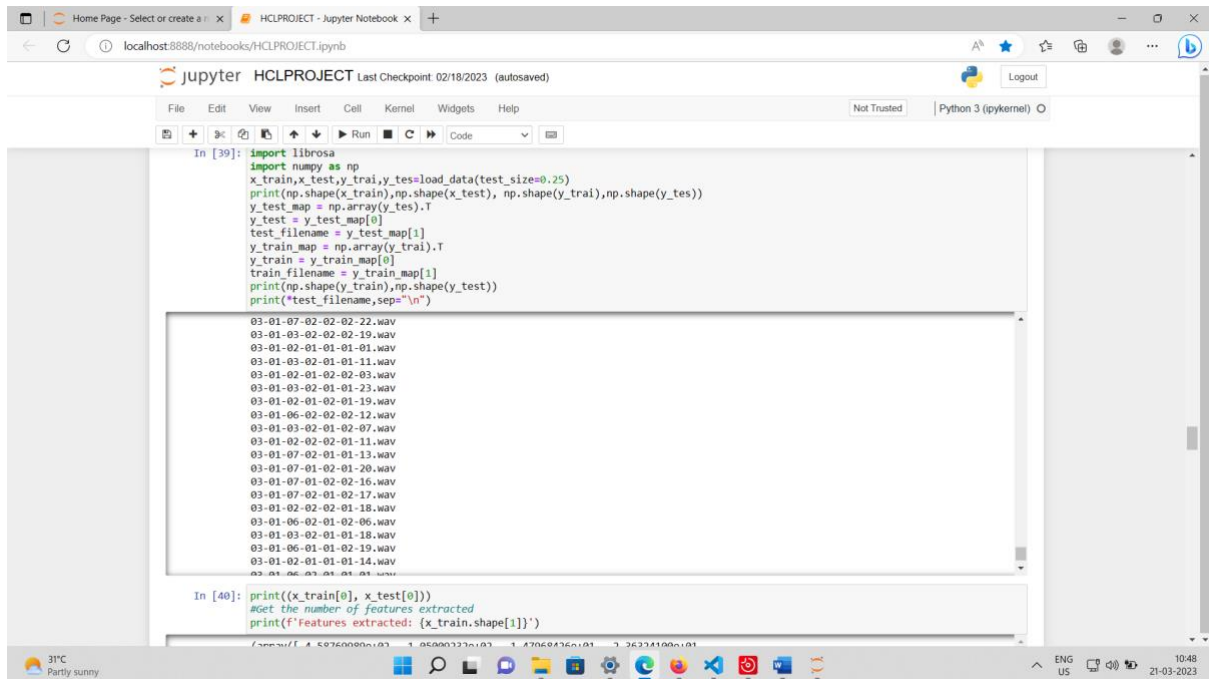
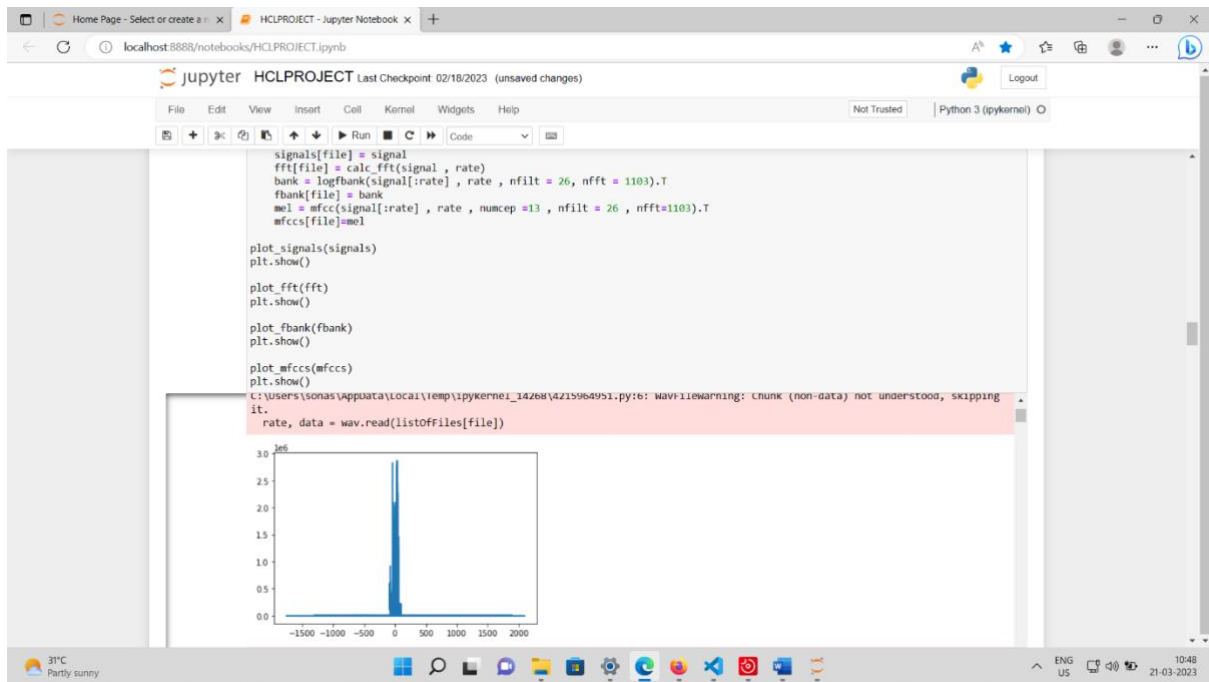
```
listoffiles = getListoffiles(dirname)
len(listoffiles)

Out[31]: 2879
```

```
In [5]: import speech_recognition as sr
r=sr.Recognizer()
for file in range(0 , len(listoffiles)):
    with sr.AudioFile(listoffiles[file]) as source:
        audio = r.listen(source)
        print(file)
        try:
            text = r.recognize_google(audio)
            print(text)
        except:
            print('error')
```

```
{
  'confidence': 0.86977035,
  'transcript': 'dog sitting by the door'},
{'confidence': 0.86977035,
 'transcript': 'dogs sitting by the door'},
{'confidence': 0.86977035,
 'transcript': 'dogs are sitting by the door'},
{'confidence': 0.86977035,
 'transcript': 'sitting by the door'}],
'final': True}
could you talking by the door
2
result2:
[{'confidence': 0.86977035,
  'transcript': 'dog sitting by the door'},
 {'confidence': 0.86977035,
  'transcript': 'dogs sitting by the door'},
 {'confidence': 0.86977035,
  'transcript': 'dogs are sitting by the door'},
 {'confidence': 0.86977035,
  'transcript': 'sitting by the door'}],
'final': True}
dog sitting by the door
3
result2:
[]
```





Home Page - Select or create a notebook | HCLPROJECT - Jupyter Notebook | localhost:8888/notebooks/HCLPROJECT.ipynb

Jupyter HCLPROJECT Last Checkpoint: 02/18/2023 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (pykernel)

03-01-02-02-01-01-19.wav
03-01-07-01-02-02-23.wav

```
In [40]: print(x_train[0], x_test[0])
#Get the number of features extracted
print(f'Features extracted: {x_train.shape[1]}')
```

```
-7.62444258e+00, -4.52498579e+00, -1.04427614e+01, -3.94952345e+00,
-6.04503775e+00, 1.09863591e+00, -4.29604864e+00, 3.33680302e-01,
9.66627896e-01, 5.17042542e+00, 5.99206543e+00, 5.98339462e+00,
5.18575966e-01, 5.20549476e-01, 5.59311867e-01, 6.99832201e-01,
7.56403327e-01, 6.64064719e-01, 5.35250405e-01, 5.61913013e-01,
5.72211266e-01, 5.79270422e-01, 5.56819916e-01, 5.63546717e-01,
3.13649398e-05, 3.01331056e-05, 1.11054243e-04, 2.10846886e-02,
8.11391547e-02, 2.02044800e-01, 2.83350855e-01, 4.43339013e-02,
6.52946299e-03, 1.45695461e-02, 2.23261844e-02, 6.20024838e-02,
1.00463368e-01, 8.67507979e-02, 6.49384856e-02, 1.24461921e-02,
2.84681376e-02, 1.30360186e-01, 7.99561143e-02, 7.25424513e-02,
6.21669702e-02, 4.20209281e-02, 1.96048685e-02, 9.85552929e-03,
1.16132628e-02, 2.97265258e-02, 3.06311455e-02, 3.16666737e-02,
3.42102461e-02, 1.39858434e-02, 1.02437986e-02, 2.55534723e-02,
6.00097030e-02, 2.12236419e-02, 2.92645022e-03, 4.96005814e-04,
1.22641702e-03, 2.81112851e-03, 4.50711325e-03, 4.00129659e-03,
2.55729631e-03, 3.45197576e-03, 1.76450808e-03, 8.26525560e-04,
3.84364091e-03, 1.33845964e-02, 6.34196447e-03, 2.39720009e-03,
1.61734654e-03, 2.01943470e-03, 1.02113178e-02, 1.11353891e-02,
6.26901072e-03, 4.76525351e-03, 4.08018271e-03, 4.32455307e-03.
```

```
In [41]: model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
```

```
In [42]: model.fit(x_train,y_train)
```

```
Out[42]: MLPClassifier(alpha=0.01, batch_size=256, hidden_layer_sizes=(300,),
learning_rate='adaptive', max_iter=500)
```

```
In [43]: #SAVING THE MODEL
```

31°C Partly sunny 10:49 21-03-2023

SOURCE CODE

```
import os
import glob
from tqdm import tqdm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
from python_speech_features import mfcc , logfbank
import librosa as lr
import os, glob, pickle
import librosa
from scipy import signal
import noisereduce as nr
from glob import glob
import librosa
get_ipython().magic('matplotlib inline')
#All the Required Packages and Libraies are installed.
import soundfile
from tensorflow.keras.layers import Conv2D,MaxPool2D, Flatten, LSTM
from keras.layers import Dropout,Dense,TimeDistributed
from keras.models import Sequential
from keras.utils import to_categorical
from sklearn.utils.class_weight import compute_class_weight
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
os.listdir(path='./speech-emotion-recognition-ravdess-data')
def getListOfFiles(dirName):
    listOfFile=os.listdir(dirName)
    allFiles=list()
    for entry in listOfFile:
        fullPath=os.path.join(dirName, entry)
        if os.path.isdir(fullPath):
            allFiles=allFiles + getListOfFiles(fullPath)
        else:
            allFiles.append(fullPath)
    return allFiles

dirName = './speech-emotion-recognition-ravdess-data'
listOfFiles = getListOfFiles(dirName)
len(listOfFiles)
import speech_recognition as sr
r=sr.Recognizer()
for file in range(0 , len(listOfFiles) , 1):
    with sr.AudioFile(listOfFiles[file]) as source:
        audio = r.listen(source)
```

```

try:
    text = r.recognize_google(audio)
    print(text)
except:
    print('error')
def envelope(y , rate, threshold):
    mask=[]
    y=pd.Series(y).apply(np.abs)
    y_mean = y.rolling(window=int(rate/10) , min_periods=1 , center = True).mean()
    for mean in y_mean:
        if mean>threshold:
            mask.append(True)
        else:
            mask.append(False)
    return mask

for file in range(0 , len(listOfFiles) , 1):
    audio , sfreq = lr.load(listOfFiles[file])
    time = np.arange(0 , len(audio))
    fig ,ax = plt.subplots()
    ax.plot(time , audio)
    ax.set(xlabel = 'Time (s)' , ylabel = 'Sound Amplitude')
    plt.show()
for file in range(0 , len(listOfFiles) , 1):
    sample_rate , samples = wavfile.read(listOfFiles[file])
    frequencies , times, spectrogram = signal.spectrogram(samples, sample_rate)
    plt.pcolormesh(times, frequencies, spectrogram)
    plt.imshow(spectrogram)
    plt.ylabel('Frequency [Hz]')
    plt.xlabel('Time [sec]')
    plt.show()
def plot_signals(signals):
    fig , axes = plt.subplots(nrows=2, ncols=5,sharex =False , sharey=True,
figsize=(20,5))
    fig.suptitle('Time Series' , size=16)
    i=0
    for x in range(2):
        for y in range(5):
            axes[x,y].set_title(list(signals.keys())[i])
            axes[x,y].plot(list(signals.values())[i])
            axes[x,y].get_xaxis().set_visible(False)
            axes[x,y].get_yaxis().set_visible(False)
            i +=1

def plot_fft(fft):
    fig , axes = plt.subplots(nrows=2, ncols=5,sharex =False , sharey=True,
figsize=(20,5))
    fig.suptitle('Fourier Transform' , size=16)
    i=0
    for x in range(2):

```

```

    for y in range(5):
        data = list(fft.values())[i]
        Y,freq = data[0] , data[1]
        axes[x,y].set_title(list(fft.keys())[i])
        axes[x,y].plot(freq , Y)
        axes[x,y].get_xaxis().set_visible(False)
        axes[x,y].get_yaxis().set_visible(False)
        i +=1

def plot_fbank(fbank):
    fig , axes = plt.subplots(nrows=2, ncols=5,sharex =False , sharey=True,
figsize=(20,5))
    fig.suptitle('Filter Bank Coefficients' , size=16)
    i=0
    for x in range(2):
        for y in range(5):
            axes[x,y].set_title(list(fbank.keys())[i])
            axes[x,y].imshow(list(fbank.values())[i],cmap='hot', interpolation = 'nearest')
            axes[x,y].get_xaxis().set_visible(False)
            axes[x,y].get_yaxis().set_visible(False)
            i +=1

def plot_mfccs(mfccs):
    fig , axes = plt.subplots(nrows=2, ncols=5,sharex =False , sharey=True,
figsize=(20,5))
    fig.suptitle('Mel Frequency Capstrum Coefficients' , size=16)
    i=0
    for x in range(2):
        for y in range(5):
            axes[x,y].set_title(list(mfccs.keys())[i])
            axes[x,y].imshow(list(mfccs.values())[i],
                                cmap='hot', interpolation = 'nearest')
            axes[x,y].get_xaxis().set_visible(False)
            axes[x,y].get_yaxis().set_visible(False)
            i +=1

def calc_fft(y,rate):
    n = len(y)
    freq = np.fft.rfftfreq(n , d= 1/rate)
    Y= abs(np.fft.rfft(y)/n)
    return(Y,freq)
import matplotlib.pyplot as plt
from scipy.io import wavfile as wav
from scipy.fftpack import fft
import numpy as np
for file in range(0 , len(listOfFiles) , 1):
    rate, data = wav.read(listOfFiles[file])
    fft_out = fft(data)
    %matplotlib inline
    plt.plot(data, np.abs(fft_out))

```

```

plt.show()

signals={}
fft={}
fbank={}
mfccs={}
# load data
for file in range(0 , len(listOfFiles) , 1):
#   rate, data = wavfile.read(listOfFiles[file])
    signal,rate =librosa.load(listOfFiles[file] , sr=44100)
    mask = envelope(signal , rate , 0.0005)
    signals[file] = signal
    fft[file] = calc_fft(signal , rate)

    bank = logfbank(signal[:rate] , rate , nfilt = 26, nfft = 1103).T
    fbank[file] = bank
    mel = mfcc(signal[:rate] , rate , numcep =13 , nfilt = 26 , nfft=1103).T
    mfccs[file]=mel

plot_signals(signals)
plt.show()

plot_fft(fft)
plt.show()

plot_fbank(fbank)
plt.show()

plot_mfccs(mfccs)
plt.show()
def envelope(y , rate, threshold):
    mask=[]
    y=pd.Series(y).apply(np.abs)
    y_mean = y.rolling(window=int(rate/10) , min_periods=1 , center = True).mean()
    for mean in y_mean:
        if mean>threshold:
            mask.append(True)
        else:
            mask.append(False)
    return mask
import glob,pickle
for file in tqdm(glob.glob(r'C:\Users\Sakshi jain\speech-emotion-recognition-ravdess-
data\**\*.wav')):
    file_name = os.path.basename(file)
    signal , rate = librosa.load(file, sr=16000)
    mask = envelope(signal,rate, 0.0005)
    wavfile.write(filename=   r'C:\Users\Sakshi   jain\clean_speech\\'+str(file_name),
rate=rate,data=signal[mask])
def extract_feature(file_name, mfcc, chroma, mel):
    with soundfile.SoundFile(file_name) as sound_file:

```

```

X = sound_file.read(dtype="float32")
sample_rate=sound_file.samplerate
if chroma:
    stft=np.abs(librosa.stft(X))
    result=np.array([])
    if mfcc:
        mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T,
axis=0)
        result=np.hstack((result, mfccs))
    if chroma:
        chroma=np.mean(librosa.feature.chroma_stft(S=stft,
sr=sample_rate).T,axis=0)
        result=np.hstack((result, chroma))
    if mel:
        mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
        result=np.hstack((result, mel))
    return result
emotions={
'01':'neutral',
'02':'calm',
'03':'happy',
'04':'sad',
'05':'angry',
'06':'fearful',
'07':'disgust',
'08':'surprised'
}
observed_emotions=['calm', 'happy', 'fearful', 'disgust']
from glob import glob
import os
import glob
def load_data(test_size=0.33):
    x,y=[],[]
    answer = 0
    for file in glob.glob(r'C:\Users\Sakshi jain\clean_speech\*.wav'):
        file_name=os.path.basename(file)
        emotion=emotions[file_name.split("-")[2]]
        if emotion not in observed_emotions:
            answer += 1
            continue
        feature=extract_feature(file, mfcc=True, chroma=True, mel=True)
        x.append(feature)
        y.append([emotion,file_name])
    return train_test_split(np.array(x), y, test_size=test_size, random_state=9)
import librosa
import numpy as np
x_train,x_test,y_train,y_test=load_data(test_size=0.25)
print(np.shape(x_train),np.shape(x_test), np.shape(y_train),np.shape(y_test))
y_test_map = np.array(y_test).T
y_test = y_test_map[0]

```

```

test_filename = y_test_map[1]
y_train_map = np.array(y_train).T
y_train = y_train_map[0]
train_filename = y_train_map[1]
print(np.shape(y_train), np.shape(y_test))
print(*test_filename, sep="\n")
print((x_train[0], x_test[0]))
print(f'Features extracted: {x_train.shape[1]}')
model = MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08,
hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500)
model.fit(x_train, y_train)
import pickle
Pkl_Filename = "Emotion_Voice_Detection_Model.pkl"
with open(Pkl_Filename, 'wb') as file:
    pickle.dump(model, file)
with open(Pkl_Filename, 'rb') as file:
    Emotion_Voice_Detection_Model = pickle.load(file)
Emotion_Voice_Detection_Model
y_pred = Emotion_Voice_Detection_Model.predict(x_test)
y_pred
import numpy as np
import pandas as pd
y_pred1 = pd.DataFrame(y_pred, columns=['predictions'])
y_pred1['file_names'] = test_filename
print(y_pred1)
y_pred1.to_csv('predictionfinal.csv')
import pyaudio
import wave
CHUNK = 1024
FORMAT = pyaudio.paInt16 #paInt8
CHANNELS = 2
RATE = 44100 #sample rate
RECORD_SECONDS = 4
WAVE_OUTPUT_FILENAME = "output10.wav"

p = pyaudio.PyAudio()

stream = p.open(format=FORMAT,
                channels=CHANNELS,
                rate=RATE,
                input=True,
                frames_per_buffer=CHUNK) #buffer

print("* recording")
frames = []

for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
    frames.append(data) # 2 bytes(16 bits) per channel

```

```

print("* done recording")

stream.stop_stream()
stream.close()
p.terminate()

wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()
#The file 'output10.wav' in the next cell is the file that was recorded live using the code
:
data, sampling_rate = librosa.load('output10.wav')
%matplotlib inline
import os
import pandas as pd
import librosa.display
import glob

plt.figure(figsize=(15, 5))
librosa.display.waveplot(data, sr=sampling_rate)
file = 'output10.wav'
ans = []
new_feature ,labels = extract_feature(file, mfcc=True, chroma=True, mel=True)
ans.append(new_feature)
ans = np.array(ans)
Emotion_Voice_Detection_Model.predict([ans])

```