UNIVERSITY OF ENGINEERING AND TECHNOLOGY TAXILA (UET)

COMPUTER SCIENCE ENGINEERING DEPARTMENT

Project

Gender Recognition System using Speech Signal

For

Fifth Semester

INSTRUCTOR: Sir. Muhammad Majid

Submitted by: Muhammad Subhan

Reg. number: 20-CP-77

Date : 09/01/2022

Table of Contents

GENDER RECOGNITION SYSTEM USING SPEECH SIGNAL

Objectives:

We are excited to embark on a journey where we will use the language of Python to create a code that could understand the nuances of the human voice. Our code will be able to distinguish between the voices of men and women with grace and precision. It's a project that blends the beauty of technology with the complexity of human communication. We can't wait to see what we will discover!

Abstract:

Signals are like whispers on the wind, carrying information and meaning as they fluctuate and change with time and place. And among all the signals that exist, none is as powerful and expressive as the human voice. Whether it's a laugh, a sob, or a simple hello, the voice is the most essential tool we have for communicating with one another. It's no wonder, then, that speech signals have become the focus of so much research, particularly in the field of Gender Recognition (GR).

GR is the process of identifying a person's gender based on their voice. It's a field that has grown in importance as technology has advanced, and it's now possible to analyze speech signals in ways that were once unimaginable. Gender, age, ethnicity, and emotional state are all factors that can be inferred from the voice, providing a wealth of information about the speaker.

The process of GR involves comparing an unknown speaker's voice sample to a database of known speakers. The recognition decision is based on the best match, and this decision allows us to determine whether the speaker is male or female. It's a fascinating process that has the potential to revolutionize the way we interact with technology. Imagine being able to talk on your phone and have it understood not just what you're saying, but who you are. The possibilities are endless.

As we continue to explore the world of speech signals and GR, one thing is certain: the voice is a powerful tool that can connect us in ways we never thought possible. And as we unlock its secrets, we'll be able to understand ourselves and each other in new and profound ways.

Introduction:

Voice recognition technology has come a long way in recent years, and it can be broadly categorized into two types: isolated and continuous. Isolated speech recognition requires a clear pause between each word, whereas continuous speech recognition can understand speech without such breaks. Another important aspect of voice recognition is whether it is speaker-dependent or speaker-independent. A speaker-dependent system is tailored to a specific individual's voice, while a speaker-independent system is able to understand a variety of voices.

In addition to these basic distinctions, voice recognition technology has a range of applications. One of the key uses of speaker recognition technology is for verification and authentication. This is when the speaker's voice is used to confirm their identity, such as when unlocking a phone or accessing a secure website. The other main application of voice recognition technology is identification, which involves using the speaker's voice to identify an unknown person.

One of the most exciting things about voice recognition technology is that it continues to evolve and improve. With advancements in machine learning and artificial intelligence, we can expect voice recognition to become even more accurate and versatile in the future.

Methodology:

Feature Extraction and Creation of Dataset:

This script is a Python script that aims to classify audio files as belonging to male or female speakers based on their speech signal. It uses several libraries, including tarfile, pandas, numpy, librosa, matplotlib, and seaborn. It does the following:

1. Extracts files from a compressed tar archive and places them in a destination folder.
2. Iterates through a directory of files, opens a specific file, and reads lines from it until it finds a line that starts with "Gender:", then checks the value of that line and moves the audio files in the corresponding folder based on the gender value.
3. Defines a function called get_mfcc, which takes in the audio signal and its sample rate as inputs, resamples the audio signal to a specific sample rate, extracts the Mel-frequency cepstral coefficients (MFCCs) of the signal, and returns these coefficients as a feature.
4. Defines a function called get_pitch, which takes in the sample rate and audio signal as inputs, calculates the pitch of the signal, and returns it as a feature.
5. Defines a function called extract_feature, which takes in the path of an audio file and its gender as inputs, extracts the MFCCs and pitch of the signal, and stores the features and the file's gender in a dataframe.
6. Extracts features from the audio files and stores them in a dataframe.
7. It utilizes two datasets: "https://www.kaggle.com/datasets/primaryobjects/voicegender" and "http://www.repository.voxforge1.org/downloads/SpeechCorpus/Trunk/Audio/Main/16kHz_16bit/".

Model Building and Evaluation and Prediction on Real Dataset:

This script is a Python script that uses machine learning algorithms to classify audio files as belonging to male or female speakers based on their speech signal features. It uses several libraries, including numpy, pandas, and various modules from scikit-learn library. It does the following:

1. Reads a csv file containing the extracted features and labels of the audio files.
2. Drops the unnecessary columns from the dataframe.
3. Splits the dataset into features and labels.
4. Converts the dataframe to a numpy array.
5. Splits the dataset into training and test sets.
6. Trains several models: Decision Tree, Random Forest, Logistic Regression, and K-Nearest Neighbors (KNN).
7. Make predictions on the test data using the trained models.
8. Prints the accuracy of the models on the test data.
9. Reads a csv file containing the extracted features of real-case audio files.
10. Prints the accuracy of the Decision Tree model on the real-case data.
11. It appears that the script is using a dataset of audio files to train and test several machine learning models, then it is using one of these models to classify real-case audio files.

The given code is a Python script that is used to create a gender recognition system using speech signals. The system uses various machine learning models such as Decision Tree, Random Forest, Logistic Regression, and K-Nearest Neighbors (KNN) to classify a person's gender based on their speech.

Data Visualization:

1. The first step is to import the necessary libraries such as Pandas, Numpy, Scikit-learn, and Plotly.
2. Next, the data is imported from a CSV file using the Pandas library. The data contains various speech signal features such as mean frequency, standard deviation, etc., along with the corresponding gender labels (male or female).
3. The data is then split into training and testing sets using the train_test_split function from the Scikit-learn library.
4. The next step is to train the various machine learning models on the training data. The Decision Tree, Random Forest, Logistic Regression, and KNN models are trained on the data using the corresponding functions from the Scikit-learn library.
5. After the models are trained, they are used to predict the gender of the individuals in the testing data. The accuracy of each model is then calculated using the accuracy_score function from the Scikit-learn library.
6. The next step is to create a data frame with the testing accuracy of all the models. The data frame is reshaped to a columnar format and then converted to a CSV file.
7. The same process is repeated for the predicting accuracy of all the models.
8. In the final step, the data from the CSV files is read using the Pandas library, and various visualizations such as bar charts, line charts, and scatter plots are created using the Plotly library to compare the accuracy of the different models.

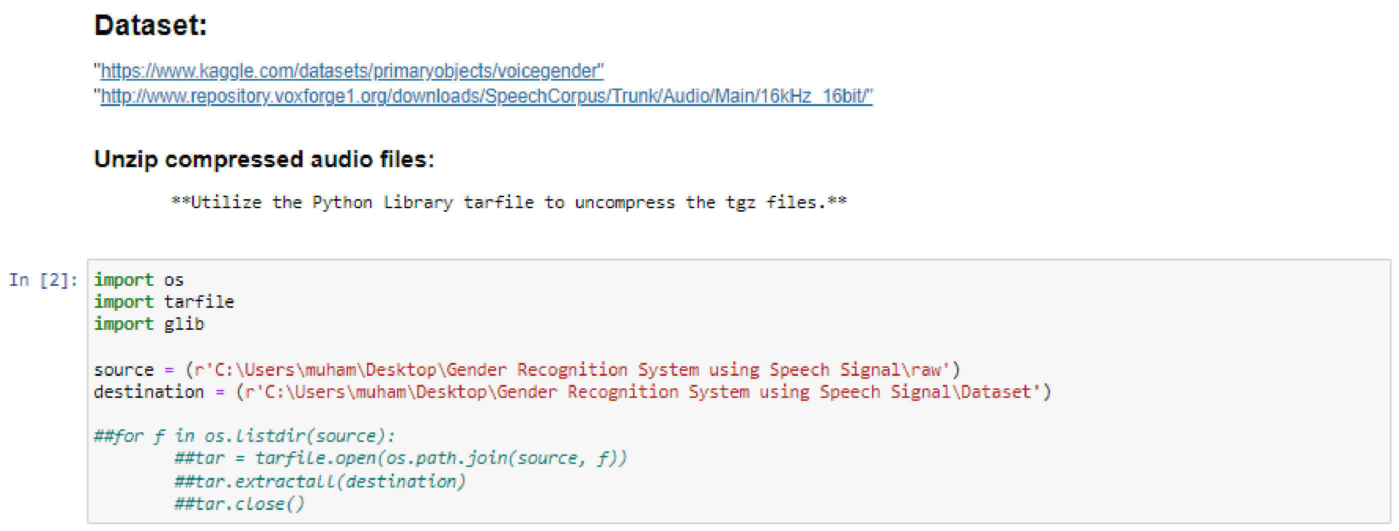
Algorithm of Code:

Here is a general algorithm step of code:

1. Import the necessary libraries such as tarfile, pandas, numpy, librosa, matplotlib, and seaborn.
2. Extract files from a compressed tar archive and place them in a destination folder.
3. Iterate through a directory of files, open a specific file, and read lines from it until it finds a line that starts with "Gender:", check the value of that line and move the audio files in the corresponding folder based on the gender value.
4. Define a function called get_mfcc, which takes in the audio signal and its sample rate as inputs, resamples the audio signal to a specific sample rate, extracts the Mel-frequency cepstral coefficients (MFCCs) of the signal, and returns these coefficients as a feature.
5. Define a function called get_pitch, which takes in the sample rate and audio signal as inputs, calculates the pitch of the signal, and returns it as a feature.
6. Define a function called extract_feature, which takes in the path of an audio file and its gender as inputs, extracts the MFCCs and pitch of the signal, and stores the features and the file's gender in a data frame.
7. Extract features from the audio filessup and store them in a data frame.
8. Read a csv file containing the extracted features and labels of the audio files.
9. Drop the unnecessary columns from the data frame.
10. Split the dataset into features and labels.
11. Convert the data frame to a NumPy array.
12. Split the dataset into training and test sets.
13. Train several models: Decision Tree, Random Forest, Logistic Regression, and K-Nearest Neighbors (KNN).
14. Predict the gender of the individuals in the testing data and calculate the accuracy of each model.
15. Print the accuracy of the models on the test data.

16. Create a data frame with the testing accuracy of all the models.
17. Convert the data frame to a CSV file.
18. Repeat the same process for the predicting accuracy of all the models.
19. In the final step, the data from the CSV files is read using the Pandas library, and various visualizations such as bar charts, line charts, and scatter plots are created using the Plotly library to compare the testing accuracy and predicting accuracy of the different models.

Screenshot of project Script:



Dataset:

"https://www.kaggle.com/datasets/primaryobjects/voicegender"
"http://www.repository.voxforge1.org/downloads/SpeechCorpus/Trunk/Audio/Main/16kHz_16bit/"

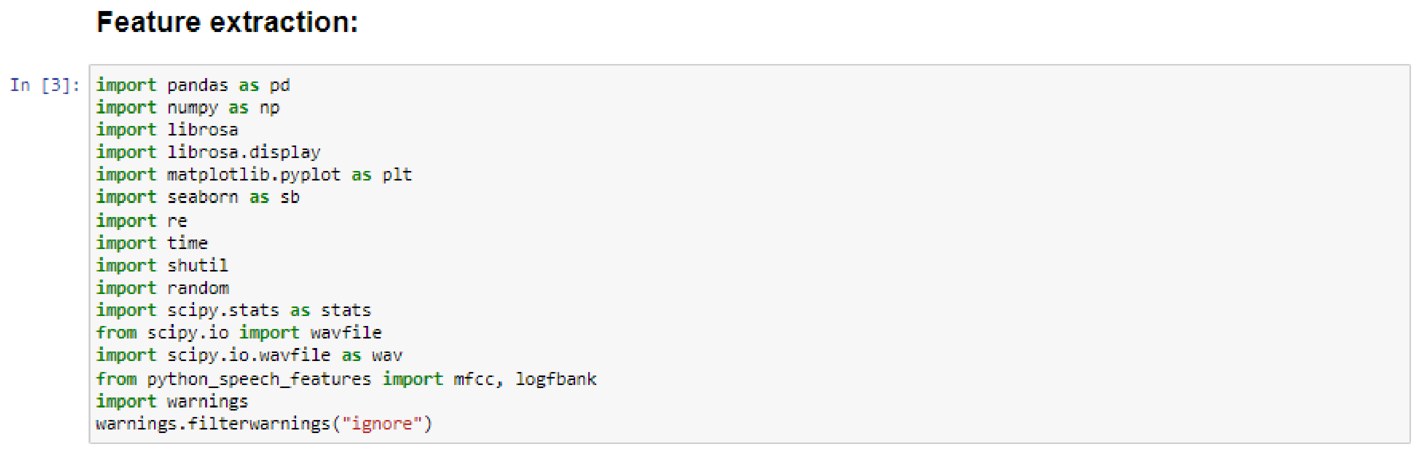
Unzip compressed audio files:

Utilize the Python Library tarfile to uncompress the tgz files.

```
In [2]: import os
        import tarfile
        import glib

        source = (r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\raw')
        destination = (r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\Dataset')

        ##for f in os.listdir(source):
                ##tar = tarfile.open(os.path.join(source, f))
                ##tar.extractall(destination)
                ##tar.close()
```

Feature Extraction and Creation of Dataset:

1. Import the necessary libraries such as tarfile, pandas, numpy, librosa, matplotlib, and seaborn.

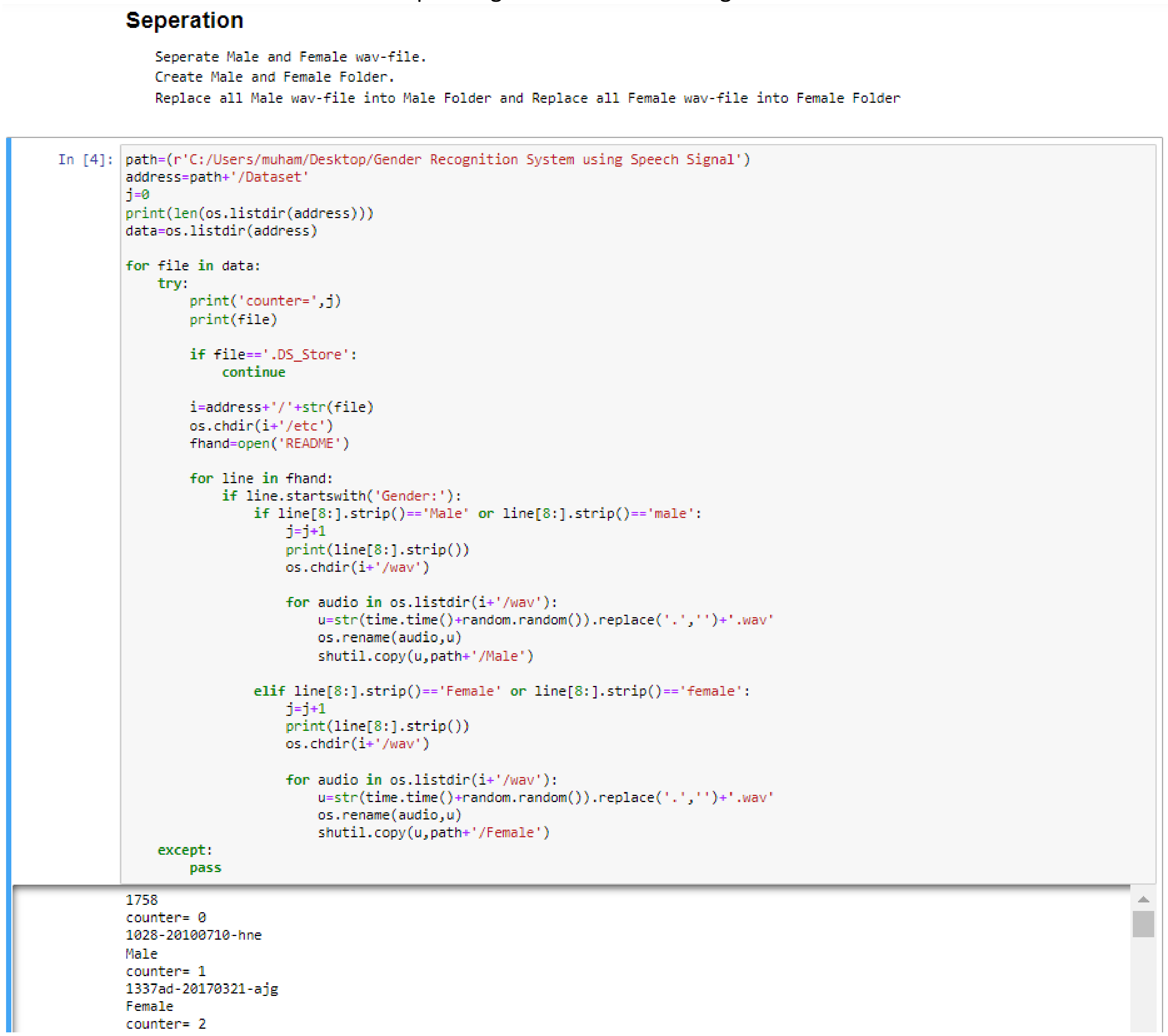


Feature extraction:

```
In [3]: import pandas as pd
        import numpy as np
        import librosa
        import librosa.display
        import matplotlib.pyplot as plt
        import seaborn as sb
        import re
        import time
        import shutil
        import random
        import scipy.stats as stats
        from scipy.io import wavfile
        import scipy.io.wavfile as wav
        from python_speech_features import mfcc, logfbank
        import warnings
        warnings.filterwarnings("ignore")
```

2. Extract files from a compressed tar archive and place them in a destination folder.

3. Iterate through a directory of files, open a specific file, and read lines from it until it finds a line that starts with "Gender:", check the value of that line and move the audio files in the corresponding folder based on the gender value.



Seperation

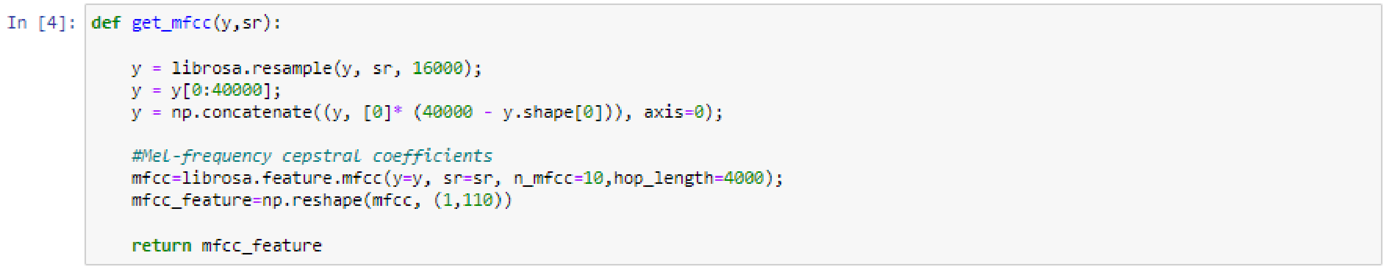
```
Seperate Male and Female wav-file.
Create Male and Female Folder.
Replace all Male wav-file into Male Folder and Replace all Female wav-file into Female Folder
```

```
In [4]: path=(r'C:/Users/muham/Desktop/Gender Recognition System using Speech Signal')
        address=path+'/Dataset'
        j=0
        print(len(os.listdir(address)))
        data=os.listdir(address)

        for file in data:
            try:
                print('counter=',j)
                print(file)

                if file=='.DS_Store':
                    continue

                i=address+'/'+str(file)
                os.chdir(i+'/etc')
                fhand=open('README')

                for line in fhand:
                    if line.startswith('Gender:'):
                        if line[8:].strip()=='Male' or line[8:].strip()=='male':
                            j=j+1
                            print(line[8:].strip())
                            os.chdir(i+'/wav')

                            for audio in os.listdir(i+'/wav'):
                                u=str(time.time()+random.random()).replace('.','')+'.wav'
                                os.rename(audio,u)
                                shutil.copy(u,path+'/Male')

                        elif line[8:].strip()=='Female' or line[8:].strip()=='female':
                            j=j+1
                            print(line[8:].strip())
                            os.chdir(i+'/wav')

                            for audio in os.listdir(i+'/wav'):
                                u=str(time.time()+random.random()).replace('.','')+'.wav'
                                os.rename(audio,u)
                                shutil.copy(u,path+'/Female')
            except:
                pass
```

```
1758
counter= 0
1028-20100710-hne
Male
counter= 1
1337ad-20170321-ajg
Female
counter= 2
```

4. Define a function called get_mfcc, which takes in the audio signal and its sample rate as inputs, resamples the audio signal to a specific sample rate, extracts the Mel-frequency cepstral coefficients (MFCCs) of the signal, and returns these coefficients as a feature.

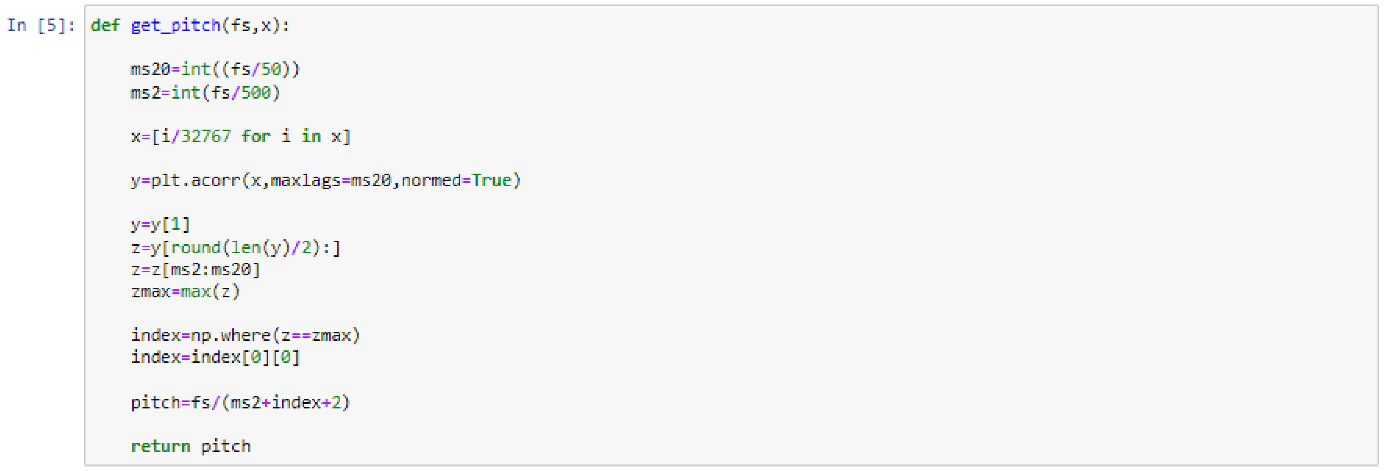
Mel-Frequency Cepstral Coefficients (MFCCs), based on some Mel-scale transformation, are a representation of the short-term power spectrum of a sound. Due to the fact that people's voices often fall within a specific frequency range and vary from person to person, it is frequently employed in speech recognition. MFCCs coefficients are employed as one of the initial steps in Speech Recognition; they identify the components of the signal and are fairly differentiating for separating the linguistic content in the signal and discarding any other information such as background noise, emotion, and so on.



```
In [4]: def get_mfcc(y,sr):

            y = librosa.resample(y, sr, 16000);
            y = y[0:40000];
            y = np.concatenate((y, [0]* (40000 - y.shape[0])), axis=0);

            #Mel-frequency cepstral coefficients
            mfcc=librosa.feature.mfcc(y=y, sr=sr, n_mfcc=10,hop_length=4000);
            mfcc_feature=np.reshape(mfcc, (1,110))

            return mfcc_feature
```

5. Define a function called get_pitch, which takes in the sample rate and audio signal as inputs, calculates the pitch of the signal, and returns it as a feature.

Pitch period is the interval of time between two successive voiced excitation cycles, or the interval of time between peaks. It is the excitation source's fundamental frequency. Therefore, an algorithm for gender categorization can employ an effective pitch extractor and a precise pitch estimation calculator.



```
In [5]: def get_pitch(fs,x):

            ms20=int((fs/50))
            ms2=int(fs/500)

            x=[i/32767 for i in x]

            y=plt.acorr(x,maxlags=ms20,normed=True)

            y=y[1]
            z=y[round(len(y)/2):]
            z=z[ms2:ms20]
            zmax=max(z)

            index=np.where(z==zmax)
            index=index[0][0]

            pitch=fs/(ms2+index+2)

            return pitch
```

6. Define a function called extract_feature, which takes in the path of an audio file and its gender as inputs, extracts the MFCCs and pitch of the signal, and stores the features and the file's gender in a data frame.

Establish a data frame with the features in it. And then store csv file.

```
In [6]: def extract_feature(path, gender):

            name_col=['name']
            pitch_col = ['pitch']
            mfcc_col = ['mfcc' + str(i + 1) for i in list(range(110))]
            columns =name_col + pitch_col + mfcc_col + ['gender']
            mydata = pd.DataFrame()

            print('Extracting features for ' + gender)

            i=0
            directory = os.listdir(path)
            for wav_file in directory:
                write_features = []

                name=wav_file
                y, sr = librosa.load(path + wav_file)
                fs, x = wav.read(path + wav_file)

                pitch = get_pitch(fs, x)
                mfcc_features = get_mfcc(y, sr)

                write_features=[name]+[pitch]+mfcc_features.tolist()[0]+[gender]

                mydata = mydata.append([write_features])

            mydata.columns = columns
            mydata.to_csv(r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\dataset_'+ gender + '.csv')
```

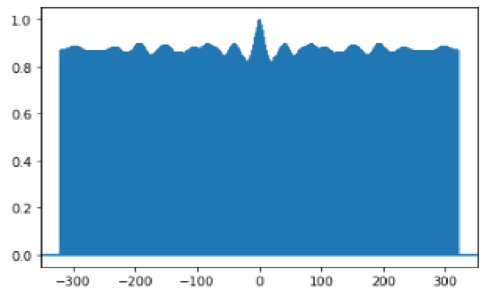
```
Extract Feature for Male (All Male Wav File feature extract and then save in csv file)
Extract Feature for Female (All Female Wav File feature extract and then save in csv file)
```

7. Extract features from the audio files and store them in a data frame.

```
Extract Feature for Male (All Male Wav File feature extract and then save in csv file)
Extract Feature for Female (All Female Wav File feature extract and then save in csv file)
```

```
In [7]: path_male=('C:/Users/muham/Desktop/Gender Recognition System using Speech Signal/Male/')
        path_female=('C:/Users/muham/Desktop/Gender Recognition System using Speech Signal/Female/')

        extract_feature(path_male,'Male')
        extract_feature(path_female,'Female')
```

```
Extracting features for Male
Extracting features for Female
```



Combine Both Male and Female csv file

```
In [172]: import pandas as pd
          femaledf = pd.read_csv(r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\dataset_female.csv')
          femaledf = femaledf.iloc[:, 1:]

          maledf = pd.read_csv(r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\dataset_male.csv')
          maledf = maledf.iloc[:, 1:]

          finaldf = femaledf.append(maledf)
          finaldf.to_csv(r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\features.csv')
```

Signal Visualization:

Visualization of Signals

In [8]:
```
sb.set_style("whitegrid", {'axes.grid' : False})
```

Specify the Male or Female Path to visualize the male and Female voices.

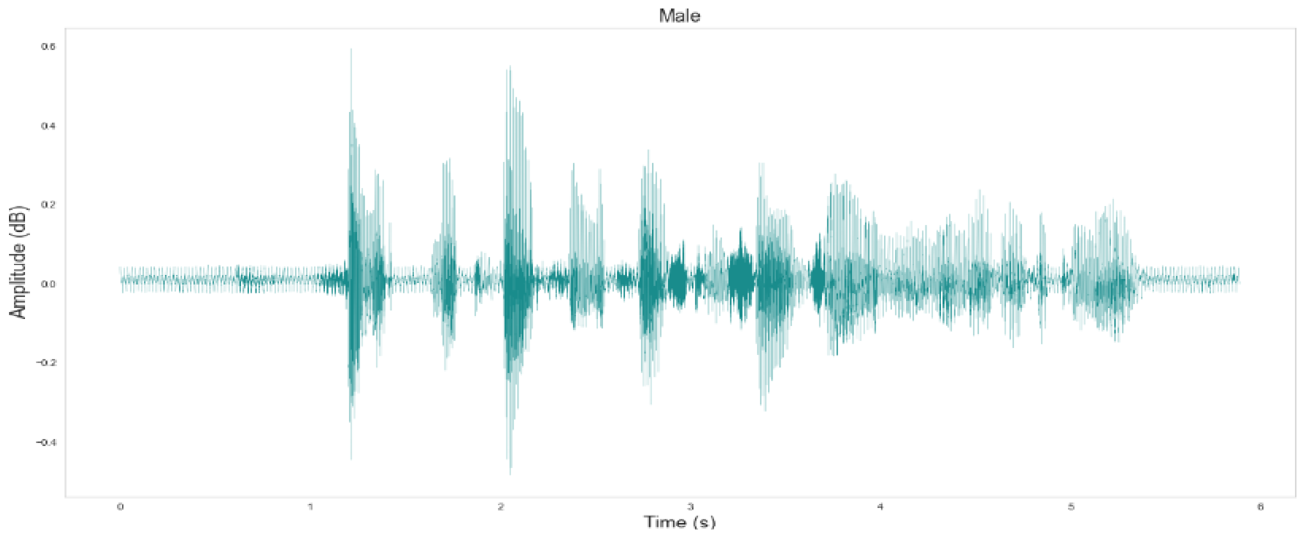
In [9]:
```
male_wav = (r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\Male\1670503968663.wav')
male_data, male_rate = librosa.load(male_wav)

female_wav = (r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\Female\1671973135257.wav')
female_data, female_rate = librosa.load(female_wav)
```

Let Start

Time Domain

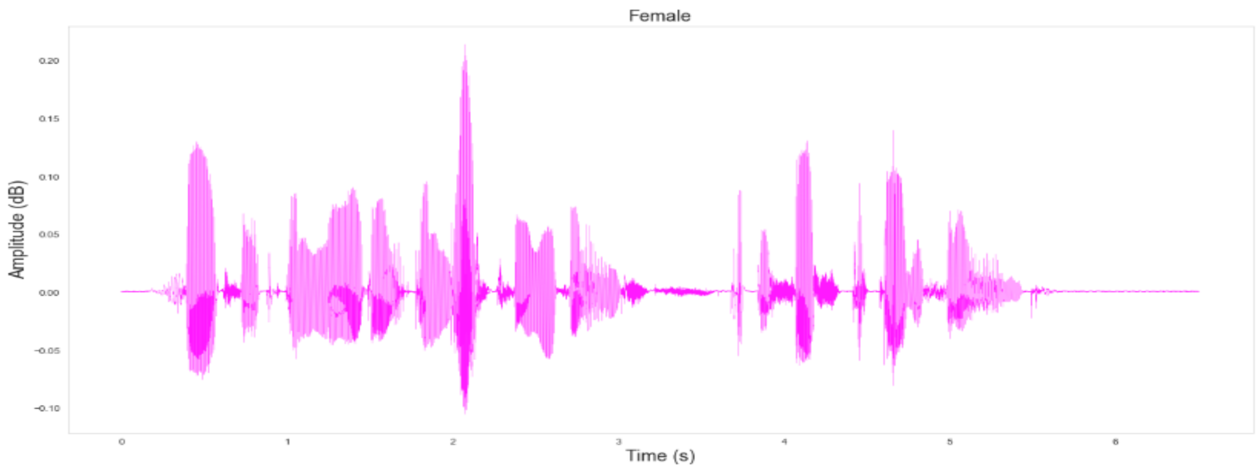
Male Signal in Time Domain

In [10]:
```
time = np.arange(0, float(male_data.shape[0]), 1) / male_rate
plt.figure(1, figsize=(20,9))
plt.subplot(111)
plt.plot(time, male_data, linewidth=0.2, alpha=0.9, color='teal') #
plt.xlabel('Time (s)', fontsize=18)
plt.ylabel('Amplitude (dB)', fontsize=18)
plt.title('Male', fontsize=18)
plt.show()
```



Female Signal in Time Domain

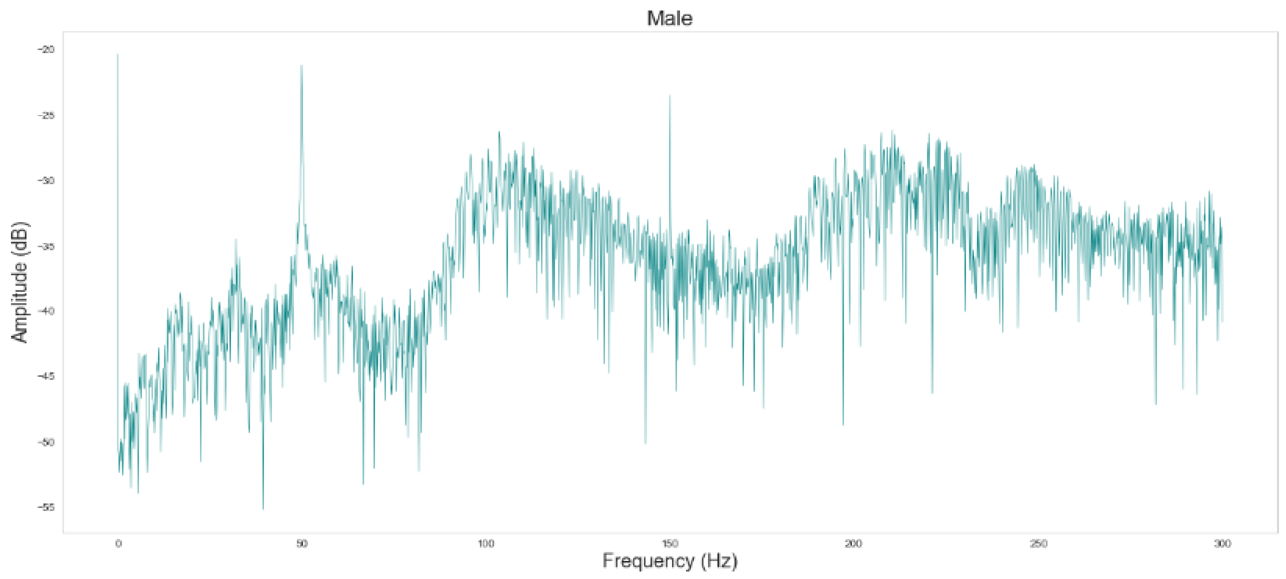
In [11]:
```
time = np.arange(0, float(female_data.shape[0]), 1) / female_rate
plt.figure(1, figsize=(20,9))
plt.subplot(111)
plt.plot(time, female_data, linewidth=0.2, alpha=0.9, color='magenta')
plt.xlabel('Time (s)', fontsize=18)
plt.ylabel('Amplitude (dB)', fontsize=18)
plt.title('Female', fontsize=18)
plt.show()
```



Frequency Domain

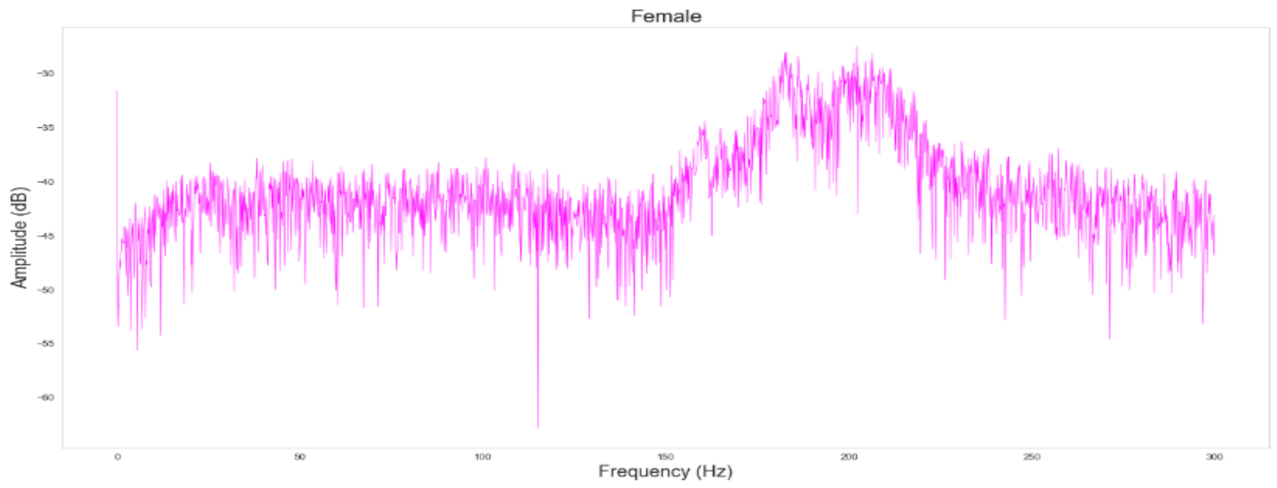
Male Signal in Frequency Domain

In [12]:
```python
fourier = np.fft.fft(male_data)
n = len(male_data)
fourier = fourier[0:int(n/2)]
# scale by the number of points so that the magnitude does not depend on the length
fourier = fourier / float(n)
#calculate the frequency at each point in Hz
freqArray = np.arange(0, (n/2), 1.0) * (male_rate*1.0/n);
x = freqArray[freqArray<300] #human voice range
y = 10*np.log10(fourier)[0:len(x)]

plt.figure(1,figsize=(20,9))
plt.plot(x, y, color='teal', linewidth=0.5)
plt.xlabel('Frequency (Hz)', fontsize=18)
plt.ylabel('Amplitude (dB)', fontsize=18)
plt.title('Male', fontsize=20)
plt.show()
```



Female Signal in Frequency Domain

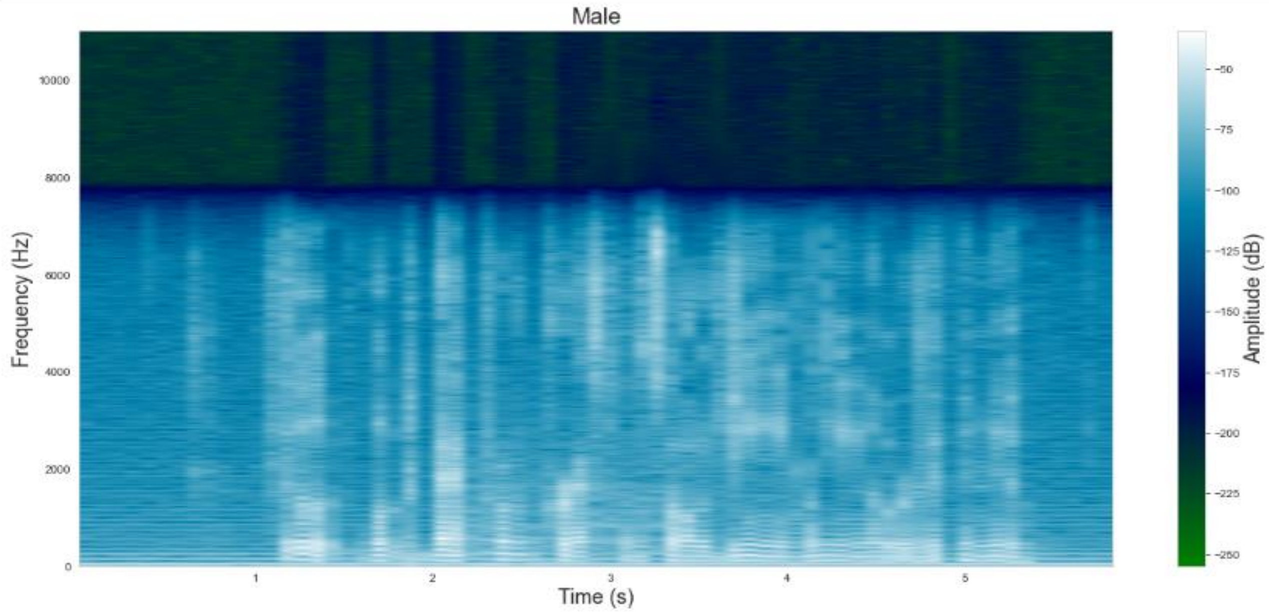
In [13]:
```python
fourier = np.fft.fft(female_data)
n = len(female_data)
fourier = fourier[0:int(n/2)]
# scale by the number of points so that the magnitude does not depend on the length
fourier = fourier / float(n)
#calculate the frequency at each point in Hz
freqArray = np.arange(0, (n/2), 1.0) * (female_rate*1.0/n);
x = freqArray[freqArray<300] #human voice range
y = 10*np.log10(fourier)[0:len(x)]

plt.figure(1,figsize=(20,9))
plt.plot(x, y, color='magenta', linewidth=0.5)
plt.xlabel('Frequency (Hz)', fontsize=18)
plt.ylabel('Amplitude (dB)', fontsize=18)
plt.title('Female', fontsize=20)
plt.show()
```



Time and Frequency Domain

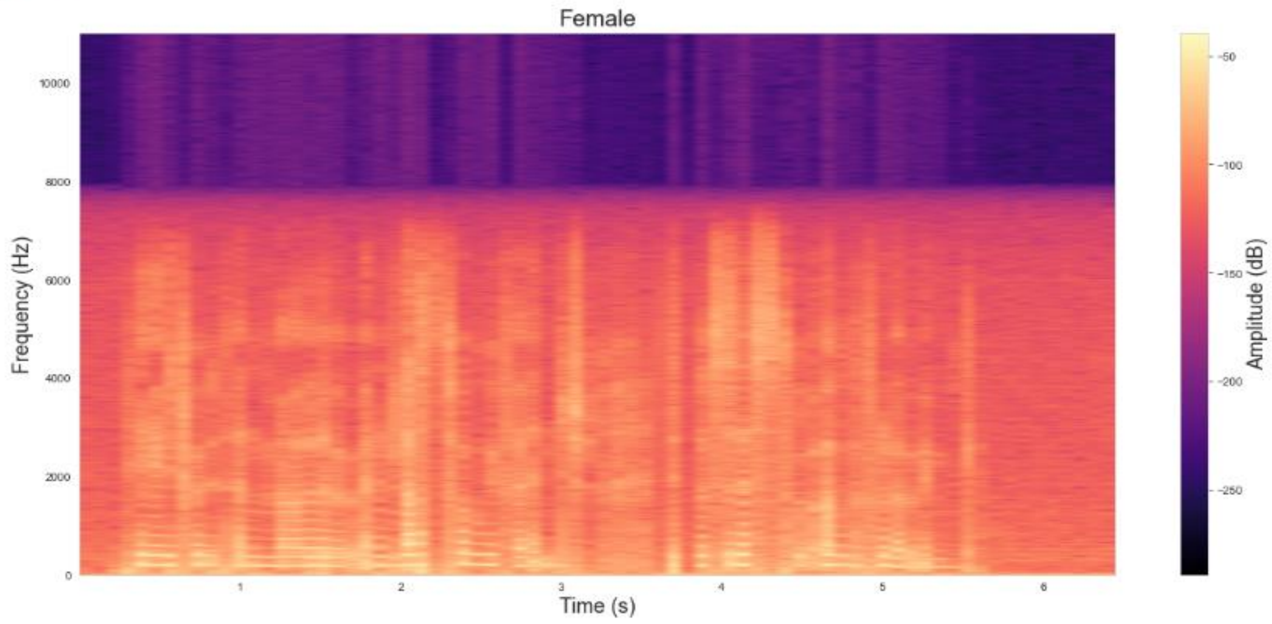
Male Signal in Time and Frequency Domain

```
In [14]: plt.figure(1,figsize=(20,9))
         plt.subplot(111)

         Pxx, freqs, bins, im = plt.specgram(male_data, Fs=male_rate, NFFT=2048, cmap=plt.get_cmap('ocean'))
         cbar=plt.colorbar(im)
         plt.xlabel('Time (s)', fontsize=18)
         plt.ylabel('Frequency (Hz)', fontsize=18)
         cbar.set_label('Amplitude (dB)', fontsize=18)
         plt.title('Male', fontsize=20)
         plt.show()
```



Female Signal in Time and Frequency Domain

```
In [15]: plt.figure(1,figsize=(20,9))
         plt.subplot(111)

         Pxx, freqs, bins, im = plt.specgram(female_data, Fs=female_rate, NFFT=2048, cmap=plt.get_cmap('magma'))
         cbar=plt.colorbar(im)
         plt.xlabel('Time (s)', fontsize=18)
         plt.ylabel('Frequency (Hz)', fontsize=18)
         cbar.set_label('Amplitude (dB)', fontsize=18)
         plt.title('Female', fontsize=20)
         plt.show()
```

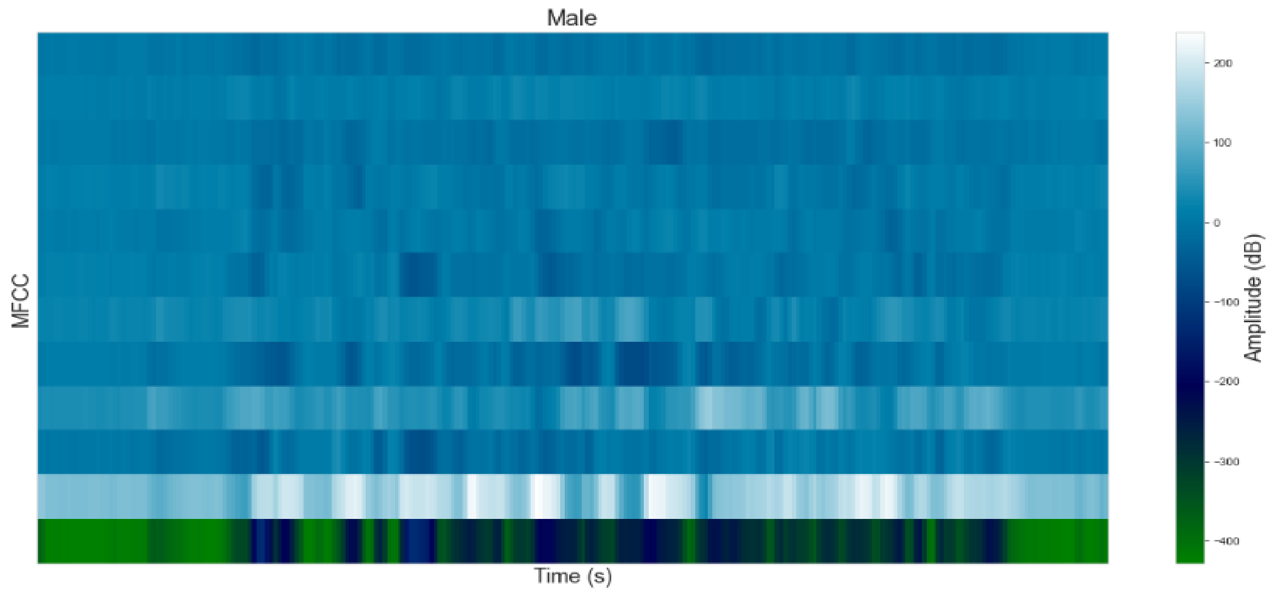


MFCC Feature

MFCCs coefficients are employed as one of the initial steps in Speech Recognition; they identify the components of the signal and are fairly differentiating for separating the linguistic content in the signal and discarding any other information such as background noise, emotion, and so on. The small squares in this graph represent MFFCs coefficients with regard to time. We can see the difference in terms of both genders; they are not identical, proving that there is a difference. The MFCC is zero in the exact location as the Male speaker stopped speaking, indicating that MFCCs do indeed selectively choose liguistic content.

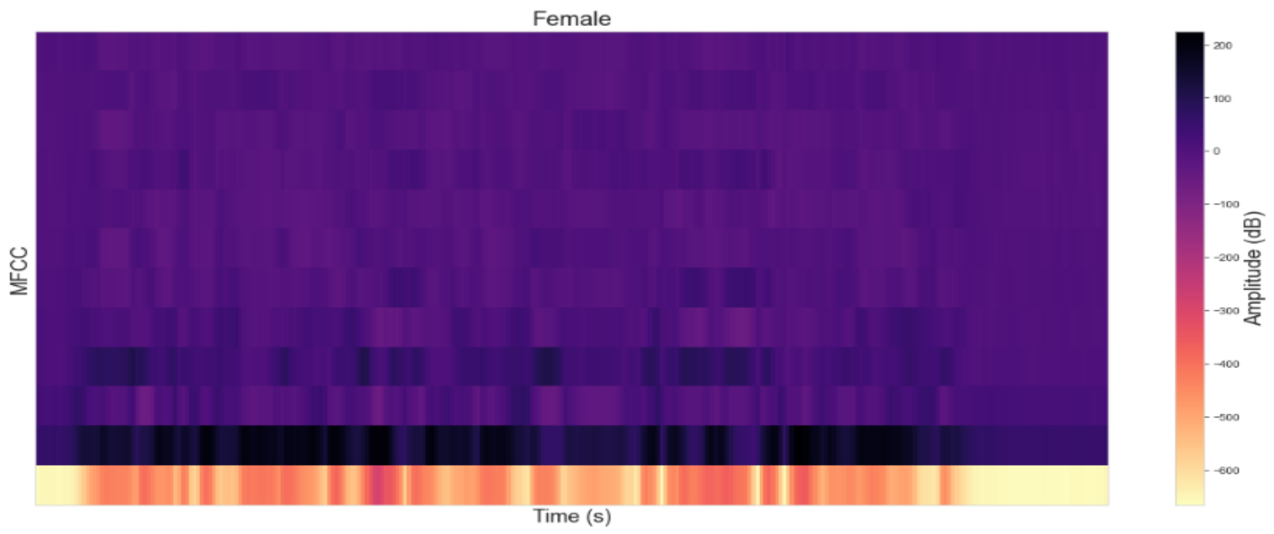
Male signal in MFCCs coefficient

```
In [16]: plt.figure(1,figsize=(20,9))
         plt.subplot(111)

         n_fft=2048
         male_mfcc = librosa.feature.mfcc(male_data, n_fft=n_fft , n_mfcc=12) #10 so we can notice the coeffecients in the graph
         im=librosa.display.specshow(male_mfcc, sr=male_rate, cmap=plt.get_cmap('ocean'))
         cbar=plt.colorbar(im)
         plt.xlabel('Time (s)', fontsize=18)
         plt.ylabel('MFCC', fontsize=18)
         cbar.set_label('Amplitude (dB)', fontsize=18)
         plt.title('Male', fontsize=20)
         plt.show()
```



Female signal in MFCCs coefficient

```
In [17]: plt.figure(1,figsize=(20,9))
         plt.subplot(111)

         female_mfcc = librosa.feature.mfcc(female_data, n_fft=n_fft , n_mfcc=12) #10 so we can notice the coeffecients in the graph
         im=librosa.display.specshow(female_mfcc, sr=female_rate, cmap=plt.get_cmap('magma_r'))
         cbar=plt.colorbar(im)
         plt.xlabel('Time (s)', fontsize=18)
         plt.ylabel('MFCC', fontsize=18)
         cbar.set_label('Amplitude (dB)', fontsize=18)
         plt.title('Female', fontsize=20)
         plt.show()
```



Audio Recording

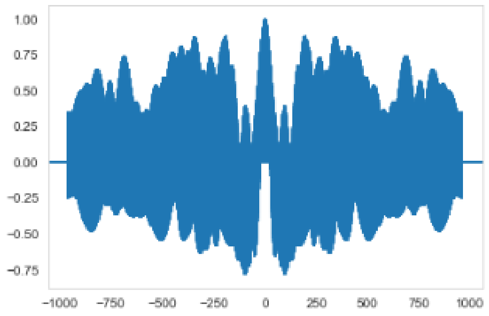
In [18]:
```python
import pyaudio
import wave
```

In [19]:
```python

def record(path):

    FORMAT = pyaudio.paInt16
    CHANNELS = 1
    RATE = 22050
    CHUNK = 1024
    RECORD_SECONDS = 5
    WAVE_OUTPUT_FILENAME = input('Enter filename:')

    audio = pyaudio.PyAudio()
    stream = audio.open(format=FORMAT, channels=CHANNELS,
                    rate=RATE, input=True,
                    frames_per_buffer=CHUNK)

    print("Recording start")

    frames = []
    for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
        data = stream.read(CHUNK)
        frames.append(data)

    print("Recording End")

    stream.stop_stream()
    stream.close()
    audio.terminate()

    waveFile = wave.open(path+'\\'+WAVE_OUTPUT_FILENAME, 'wb')
    waveFile.setnchannels(CHANNELS)
    waveFile.setsampwidth(audio.get_sample_size(FORMAT))
    waveFile.setframerate(RATE)
    waveFile.writeframes(b''.join(frames))
    waveFile.close()
```

Extracting Features for recorded audios

In [20]:
```python
path_recorded=(r'C:/Users/muham/Desktop/Gender Recognition System using Speech Signal/Recorded/')
extract_feature(path_recorded,'recorded')
```

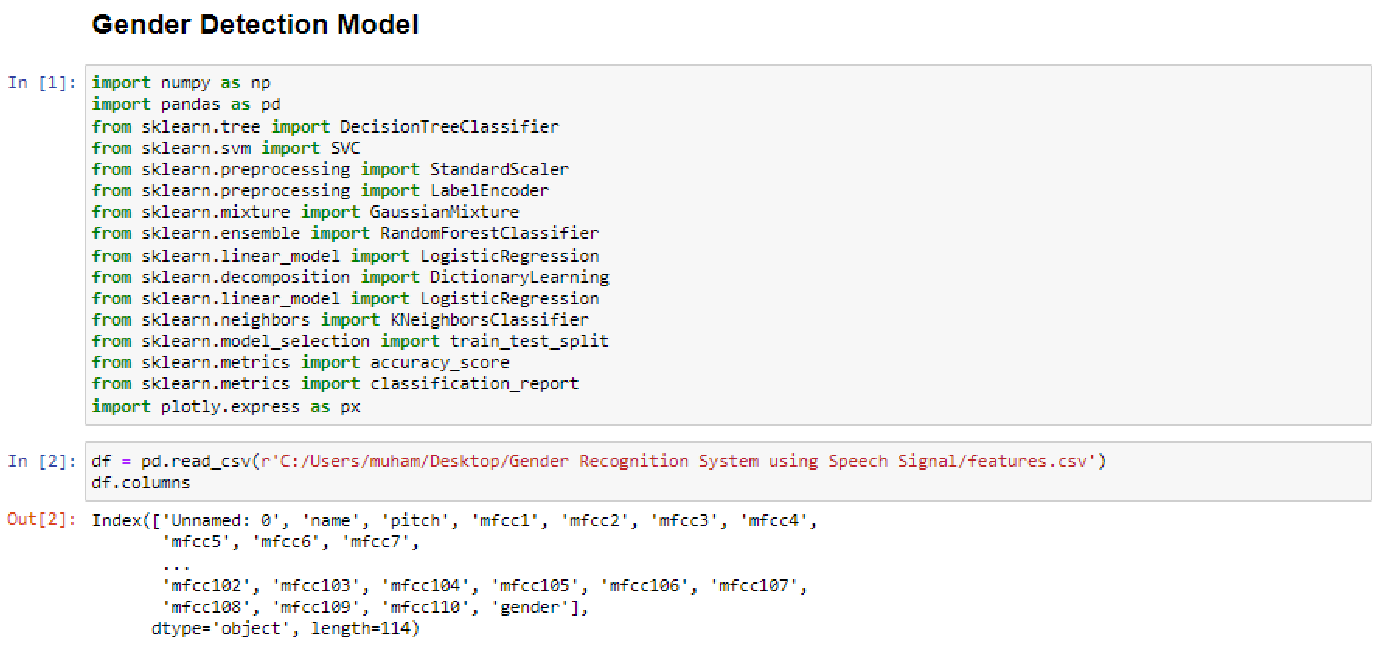
```
Extracting features for recorded
```



In [21]:
```python
df = pd.read_csv(r'C:/Users/muham/Desktop/Gender Recognition System using Speech Signal/dataset_recorded.csv')

df['gender']=' '
df.to_csv(r'C:/Users/muham/Desktop/Gender Recognition System using Speech Signal/dataset_recorded.csv', index=False)
```

Model Building and Evaluation and Prediction on Testand Real Dataset:

1. Read a csv file containing the extracted features and labels of the audio files.

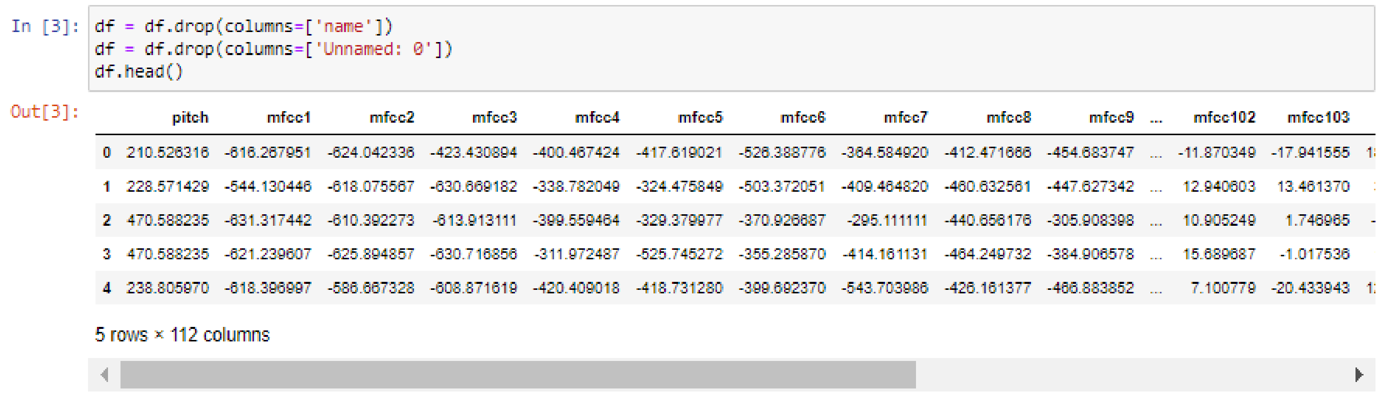


Gender Detection Model

```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.svm import SVC
        from sklearn.preprocessing import StandardScaler
        from sklearn.preprocessing import LabelEncoder
        from sklearn.mixture import GaussianMixture
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.linear_model import LogisticRegression
        from sklearn.decomposition import DictionaryLearning
        from sklearn.linear_model import LogisticRegression
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import classification_report
        import plotly.express as px
```

```
In [2]: df = pd.read_csv(r'C:/Users/muham/Desktop/Gender Recognition System using Speech Signal/features.csv')
        df.columns

Out[2]: Index(['Unnamed: 0', 'name', 'pitch', 'mfcc1', 'mfcc2', 'mfcc3', 'mfcc4',
               'mfcc5', 'mfcc6', 'mfcc7',
               ...
               'mfcc102', 'mfcc103', 'mfcc104', 'mfcc105', 'mfcc106', 'mfcc107',
               'mfcc108', 'mfcc109', 'mfcc110', 'gender'],
              dtype='object', length=114)
```

2. Drop the unnecessary columns from the data frame.



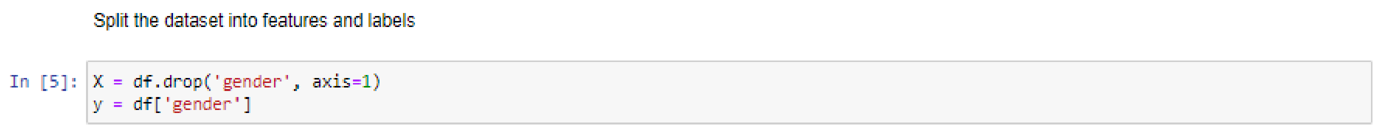
```
In [3]: df = df.drop(columns=['name'])
        df = df.drop(columns=['Unnamed: 0'])
        df.head()
```

Out[3]:

	pitch	mfcc1	mfcc2	mfcc3	mfcc4	mfcc5	mfcc6	mfcc7	mfcc8	mfcc9	...	mfcc102	mfcc103
0	210.526316	-616.267951	-624.042336	-423.430894	-400.467424	-417.619021	-526.388776	-364.584920	-412.471666	-454.683747	...	-11.870349	-17.941555
1	228.571429	-544.130446	-618.075567	-630.669182	-338.782049	-324.475849	-503.372051	-409.464820	-460.632561	-447.627342	...	12.940603	13.461370
2	470.588235	-631.317442	-610.392273	-613.913111	-399.559464	-329.379977	-370.926687	-295.111111	-440.656176	-305.908398	...	10.905249	1.746965
3	470.588235	-621.239607	-625.894857	-630.716856	-311.972487	-525.745272	-355.285870	-414.161131	-464.249732	-384.906578	...	15.689687	-1.017536
4	238.805970	-618.396997	-586.667328	-608.871619	-420.409018	-418.731280	-399.692370	-543.703986	-426.161377	-466.883852	...	7.100779	-20.433943

5 rows × 112 columns

3. Split the dataset into features and labels.



Split the dataset into features and labels

```
In [5]: X = df.drop('gender', axis=1)
        y = df['gender']
```

4. Convert the data frame to a NumPy array.

Convert the DataFrame to a numpy array

```
In [6]: X = X.values
        y = y.values
```

```
In [7]: X
```

```
Out[7]: array([[ 210.5263158 , -616.2679514 , -624.0423356 , ...,   -6.45537661,
           16.98404011,   10.60812415],
         [ 228.5714286 , -544.1304457 , -618.0755671 , ...,   11.46724739,
          -13.3121816 ,   -0.79591977],
         [ 470.5882353 , -631.3174417 , -610.3922729 , ...,   -3.4313383 ,
           11.37681534,    5.16055332],
         ...,
         [ 122.1374046 , -687.8178305 , -688.0670366 , ...,  -13.88873406,
          -16.44806761,   -8.11357495],
         [ 250.        , -663.6538734 , -666.9625189 , ...,  -17.71910609,
          -29.63814727,    0.        ],
         [ 114.2857143 , -354.3246838 , -310.4279053 , ...,  -16.44407338,
          -11.99945919,    0.        ]])
```

```
In [8]: y
```

```
Out[8]: array(['Female', 'Female', 'Female', ..., 'Male', 'Male', 'Male'],
              dtype=object)
```

5. Split the dataset into training and test sets.

Split the dataset into training and test sets

```
In [9]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

6. Train several models: Decision Tree, Random Forest, Logistic Regression, and K-Nearest Neighbors (KNN).
7. Predict the gender of the individuals in the testing data and real data and calculate the accuracy of each model.
8. Print the accuracy of the models on the test data and real data.

Recognize Gender By using Decision Tree Model

As the name implies, a decision tree is a flow-like tree structure that operates on the principle of conditions. It is effective and includes powerful algorithms for predictive analysis. It primarily consists of internal nodes, branches, and a terminal node. Every internal node represents a "test" on an attribute, whereas branches represent the test's conclusion and every leaf node represents the class name. When it comes to supervised learning approaches, this is the most often used algorithm.

Create a decision tree classifier

```
In [10]: model = DecisionTreeClassifier()
```

Train the classifier on the training data

```
In [11]: model.fit(X_train, y_train)
```

```
Out[11]: DecisionTreeClassifier()
```

Make predictions on the test data

```
In [12]: y_pred = model.predict(X_test)
```

Print the accuracy

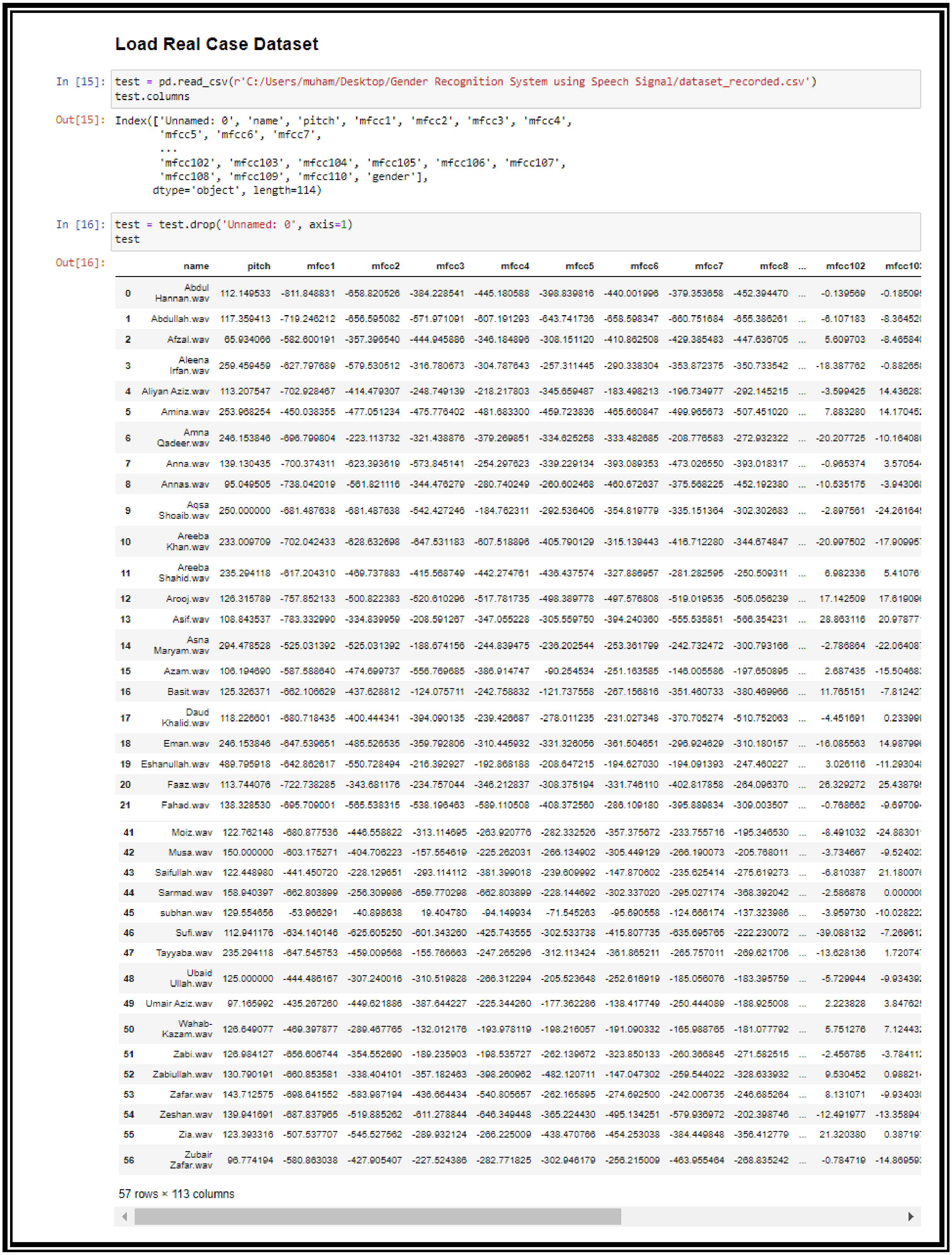
```
In [13]: decision_accuracy = accuracy_score(y_test, y_pred)
         print("Accuracy: ", decision_accuracy)
```

```
Accuracy:  0.9104922923918448
```

Evaluate the model's performance

```
In [14]: report = classification_report(y_test, y_pred)
         print(report)
```

```
              precision    recall  f1-score   support

      Female       0.82      0.86      0.84      2713
        Male       0.95      0.93      0.94      7342

    accuracy                           0.91     10055
   macro avg       0.88      0.89      0.89     10055
weighted avg       0.91      0.91      0.91     10055
```



Load Real Case Dataset

```
In [15]: test = pd.read_csv(r'C:/Users/muham/Desktop/Gender Recognition System using Speech Signal/dataset_recorded.csv')
         test.columns
```

```
Out[15]: Index(['Unnamed: 0', 'name', 'pitch', 'mfcc1', 'mfcc2', 'mfcc3', 'mfcc4',
                'mfcc5', 'mfcc6', 'mfcc7',
                ...
                'mfcc102', 'mfcc103', 'mfcc104', 'mfcc105', 'mfcc106', 'mfcc107',
                'mfcc108', 'mfcc109', 'mfcc110', 'gender'],
               dtype='object', length=114)
```

```
In [16]: test = test.drop('Unnamed: 0', axis=1)
         test
```

Out[16]:

	name	pitch	mfcc1	mfcc2	mfcc3	mfcc4	mfcc5	mfcc6	mfcc7	mfcc8	...	mfcc102	mfcc10
0	Abdul Hannan.wav	112.149533	-811.848831	-658.820526	-384.228541	-445.180588	-398.839816	-440.001996	-379.353658	-452.394470	...	-0.139569	-0.18509
1	Abdullah.wav	117.359413	-719.246212	-656.595082	-571.971091	-607.191293	-643.741736	-658.598347	-660.751684	-655.386261	...	-6.107183	-8.36452
2	Afzal.wav	65.934066	-582.600191	-357.396540	-444.945886	-346.184896	-308.151120	-410.862508	-429.385483	-447.636705	...	5.609703	-8.46584
3	Aleena Irfan.wav	259.459459	-627.797689	-579.530512	-316.780673	-304.787643	-257.311445	-290.338304	-353.872375	-350.733542	...	-18.387762	-0.88265
4	Aliyan Aziz.wav	113.207547	-702.928467	-414.479307	-248.749139	-218.217803	-345.659487	-183.498213	-196.734977	-292.145215	...	-3.599425	14.43628
5	Amina.wav	253.968254	-450.038355	-477.051234	-475.776402	-481.683300	-459.723836	-465.660847	-499.965673	-507.451020	...	7.883280	14.17045
6	Amna Qadeer.wav	246.153846	-696.799804	-223.113732	-321.438876	-379.269851	-334.625258	-333.482685	-208.776583	-272.932322	...	-20.207725	-10.16408
7	Anna.wav	139.130435	-700.374311	-623.393619	-573.845141	-254.297623	-339.229134	-393.089353	-473.026550	-393.018317	...	-0.965374	3.57054
8	Annas.wav	95.049505	-738.042019	-561.821116	-344.476279	-280.740249	-260.602468	-460.672637	-375.568225	-452.192380	...	-10.535175	-3.94306
9	Aqsa Shoaib.wav	250.000000	-681.487638	-681.487638	-542.427246	-184.762311	-292.536406	-354.819779	-335.151364	-302.302683	...	-2.897561	-24.26164
10	Areeba Khan.wav	233.009709	-702.042433	-628.632698	-647.531183	-607.518896	-405.790129	-315.139443	-416.712280	-344.674847	...	-20.997502	-17.90995
11	Areeba Shahid.wav	235.294118	-617.204310	-469.737883	-415.568749	-442.274761	-436.437574	-327.886957	-281.282595	-250.509311	...	6.982336	5.41076
12	Arooj.wav	126.315789	-757.852133	-500.822383	-520.610296	-517.781735	-498.389778	-497.576808	-519.019535	-505.056239	...	17.142509	17.61909
13	Asif.wav	108.843537	-783.332990	-334.839959	-208.591267	-347.055228	-305.559750	-394.240360	-555.535851	-566.354231	...	28.863116	20.97877
14	Asna Maryam.wav	294.478528	-525.031392	-525.031392	-188.674156	-244.839475	-236.202544	-253.361799	-242.732472	-300.793166	...	-2.786864	-22.06408
15	Azam.wav	106.194690	-587.588640	-474.699737	-556.769685	-386.914747	-90.254534	-251.163585	-146.005586	-197.650895	...	2.687435	-15.50468
16	Basit.wav	125.326371	-662.106629	-437.628812	-124.075711	-242.758832	-121.737558	-267.156816	-351.460733	-380.469966	...	11.765151	-7.81242
17	Daud Khalid.wav	118.226601	-680.718435	-400.444341	-394.090135	-239.426687	-278.011235	-231.027348	-370.705274	-510.752063	...	-4.451691	0.23399
18	Eman.wav	246.153846	-647.539651	-485.526535	-359.792806	-310.445932	-331.326056	-361.504651	-296.924629	-310.180157	...	-16.085563	14.98799
19	Eshanullah.wav	489.795918	-642.862617	-550.728494	-216.392927	-192.868188	-208.647215	-194.627030	-194.091393	-247.460227	...	3.026116	-11.29304
20	Faaz.wav	113.744076	-722.738285	-343.681176	-234.757044	-346.212837	-308.375194	-331.746110	-402.817858	-264.096370	...	26.329272	25.43879
21	Fahad.wav	138.328530	-695.709001	-565.538315	-538.196463	-589.110508	-408.372560	-286.109180	-395.889834	-309.003507	...	-0.768662	-9.69709
41	Moiz.wav	122.762148	-680.877536	-446.558822	-313.114695	-263.920776	-282.332526	-357.375672	-233.755716	-195.346530	...	-8.491032	-24.88301
42	Musa.wav	150.000000	-603.175271	-404.706223	-157.554619	-225.262031	-266.134902	-305.449129	-266.190073	-205.768011	...	-3.734667	-9.52402
43	Saifullah.wav	122.448980	-441.450720	-228.129651	-293.114112	-381.399018	-239.609992	-147.870602	-235.625414	-275.619273	...	-6.810387	21.18007
44	Sarmad.wav	158.940397	-662.803899	-256.309986	-659.770298	-662.803899	-228.144692	-302.337020	-295.027174	-368.392042	...	-2.586878	0.00000
45	subhan.wav	129.554656	-53.966291	-40.898638	19.404780	-94.149934	-71.545263	-95.690558	-124.666174	-137.323986	...	-3.959730	-10.02822
46	Sufi.wav	112.941176	-634.140146	-625.605250	-601.343260	-425.743555	-302.533738	-415.807735	-635.695765	-222.230072	...	-39.088132	-7.26961
47	Tayyaba.wav	235.294118	-647.545753	-459.009568	-155.766663	-247.265296	-312.113424	-361.865211	-265.757011	-269.621706	...	-13.628136	1.72074
48	Ubaid Ullah.wav	125.000000	-444.486167	-307.240016	-310.519828	-266.312294	-205.523648	-252.616919	-185.056076	-183.395759	...	-5.729944	-9.93439
49	Umair Aziz.wav	97.165992	-435.267260	-449.621886	-387.644227	-225.344260	-177.362286	-138.417749	-250.444089	-188.925008	...	2.223828	3.84762
50	Wahab-Kazam.wav	126.649077	-469.397877	-289.467765	-132.012176	-193.978119	-198.216057	-191.090332	-165.988765	-181.077792	...	5.751276	7.12443
51	Zabi.wav	126.984127	-656.606744	-354.552690	-189.235903	-198.535727	-262.139672	-323.850133	-260.366845	-271.582515	...	-2.456785	-3.78411
52	Zabiullah.wav	130.790191	-660.853581	-338.404101	-357.182463	-398.260962	-482.120711	-147.047302	-259.544022	-328.633932	...	9.530452	0.98821
53	Zafar.wav	143.712575	-698.641552	-583.987194	-436.664434	-540.805657	-262.165895	-274.692500	-242.006735	-246.685264	...	8.131071	-9.93403
54	Zeshan.wav	139.941691	-687.837965	-519.885262	-611.278844	-646.349448	-365.224430	-495.134251	-579.936972	-202.398746	...	-12.491977	-13.35894
55	Zia.wav	123.393316	-507.537707	-545.527562	-289.932124	-266.225009	-438.470766	-454.253038	-384.449848	-356.412779	...	21.320380	0.38719
56	Zubair Zafar.wav	96.774194	-580.863038	-427.905407	-227.524386	-282.771825	-302.946179	-256.215009	-463.955464	-268.835242	...	-0.784719	-14.86959

57 rows × 113 columns

```
In [17]: test = test.drop(columns=['name'])
```

Test Model on Real Case using decision tree model

```
In [18]: test1 = test.drop('gender', axis=1)
         test1 = test1.values
```

```
In [19]: pred = model.predict(test1)
```

```
In [20]: pred
```

```
Out[20]: array(['Male', 'Male', 'Male', 'Female', 'Male', 'Female', 'Male', 'Male',
                'Male', 'Male', 'Female', 'Female', 'Male', 'Male', 'Female',
                'Male', 'Male', 'Male', 'Male', 'Female', 'Male', 'Male', 'Female',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Female',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Female'], dtype=object)
```

```
In [21]: test['gender'].values
```

```
Out[21]: array(['Male', 'Male', 'Male', 'Female', 'Male', 'Female', 'Female',
                'Male', 'Male', 'Female', 'Female', 'Female', 'Female', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Female', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Female', 'Female', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male'], dtype=object)
```

```
In [22]: dt_pred_accuracy = np.mean(pred == test['gender'].values)
         print("Decision Tree model Accuracy: ", dt_pred_accuracy)

         Decision Tree model Accuracy:  0.8596491228070176
```

Recognize Gender By using Random Forest Classifier Model

A random forest is a machine learning approach for solving regression and classification issues. It makes use of ensemble learning, a technique that combines several classifiers to solve complicated problems. A random forest algorithm is made up of several decision trees. The random forest algorithm's 'forest' is trained via bagging or bootstrap aggregation. Bagging is a meta-algorithm that increases the accuracy of machine learning algorithms using an ensemble approach.

Create a Random Forest model

```
In [23]: model = RandomForestClassifier(n_estimators=100)
```

Train the classifier on the training data

```
In [24]: model.fit(X_train, y_train)

Out[24]: RandomForestClassifier()
```

Make predictions on the test data

```
In [25]: y_pred = model.predict(X_test)
```

Print the accuracy

```
In [26]: random_forest_accuracy = accuracy_score(y_test, y_pred)
         print("Accuracy: ", random_forest_accuracy)

         Accuracy:  0.9529587270014918
```

Evaluate the model's performance

```
In [27]: random_forest_report = classification_report(y_test, y_pred)
         print(random_forest_report)

                       precision    recall  f1-score   support

               Female       0.93      0.89      0.91      2713
                 Male       0.96      0.98      0.97      7342

             accuracy                           0.95     10055
            macro avg       0.95      0.93      0.94     10055
         weighted avg       0.95      0.95      0.95     10055
```

Test Model on Real Case using Random Forest Classifier Model

```
In [28]: pred = model.predict(test1)
```

```
In [29]: pred
```

```
Out[29]: array(['Male', 'Male', 'Male', 'Male', 'Male', 'Female', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male'], dtype=object)
```

```
In [30]: test['gender'].values
```

```
Out[30]: array(['Male', 'Male', 'Male', 'Female', 'Male', 'Female', 'Female',
                'Male', 'Male', 'Female', 'Female', 'Female', 'Female', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Female', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Female', 'Female', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male'], dtype=object)
```

```
In [31]: RF_pred_accuracy = np.mean(pred == test["gender"].values)
         print("Random Forest Accuracy:", RF_pred_accuracy)

         Random Forest Accuracy: 0.7719298245614035
```

Recognize Gender By using Logistic Regression Model

Logistic Regression is a "Supervised Machine Learning" technique that may be used to predict the likelihood of a specific class or event. It is employed when the data is separable linearly and the outcome is binary or dichotomous. Logistic regression is a classification procedure that assigns observations to one of many classes.

Create a Logistic Regression Model

```
In [32]: model = LogisticRegression(max_iter=1000)
```

```
In [33]: from sklearn.preprocessing import StandardScaler

         scaler = StandardScaler()
         X_train_scaled = scaler.fit_transform(X_train)
         X_test_scaled = scaler.transform(X_test)
```

Train the classifier on the training data

```
In [34]: model.fit(X_train_scaled, y_train)
```

```
Out[34]: LogisticRegression(max_iter=1000)
```

Make predictions on the test data

```
In [35]: y_pred = model.predict(X_test)
```

Print the accuracy

```
In [36]: logisitc_reg_accuracy = accuracy_score(y_test, y_pred)
         print("Accuracy: ", logisitc_reg_accuracy)

         Accuracy:  0.28214818498259575
```

Evaluate the model's performance

```
In [37]: logistic_regression_report = classification_report(y_test, y_pred)
         print(logistic_regression_report)

                       precision    recall  f1-score   support

               Female       0.27      1.00      0.43      2713
                 Male       0.99      0.02      0.03      7342

             accuracy                           0.28     10055
            macro avg       0.63      0.51      0.23     10055
         weighted avg       0.80      0.28      0.14     10055
```

Test Model on Real Case using Logistic Regression Model

In [38]:
```
test_scaled = scaler.transform(test1)
```

In [39]:
```
pred = model.predict(test_scaled)
```

In [40]:
```
pred
```

Out[40]:
```
array(['Male', 'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Male',
       'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
       'Male', 'Male', 'Male', 'Female', 'Female', 'Male', 'Male', 'Male',
       'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
       'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Female',
       'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
       'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
       'Male'], dtype=object)
```

In [41]:
```
test['gender'].values
```

Out[41]:
```
array(['Male', 'Male', 'Male', 'Female', 'Male', 'Female', 'Female',
       'Male', 'Male', 'Female', 'Female', 'Female', 'Female', 'Male',
       'Female', 'Male', 'Male', 'Male', 'Female', 'Male', 'Male', 'Male',
       'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
       'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
       'Female', 'Female', 'Female', 'Male', 'Male', 'Male', 'Male',
       'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Male', 'Male',
       'Male', 'Male', 'Male', 'Male'], dtype=object)
```

In [42]:
```
LG_pred_accuracy = np.mean(pred == test["gender"].values)
print("Logistic Regression Accuracy:", LG_pred_accuracy)
```

```
Logistic Regression Accuracy: 0.7368421052631579
```

Recognize Gender By using K-Nearest Neighbours (KNN) Model

KNN is a supervised machine learning technique that may be used to solve classification and regression issues. One of the easiest algorithms to learn is K closest neighbour. K closest neighbour is a non-parametric function, i.e. It makes no assumptions about underlying data assumptions. K closest neighbour is also known as a lazy algorithm since it does not learn during the training phase, instead storing the data points and learning during the testing phase.

Create a K-Nearest Neighbours (KNN) Model

In [43]:
```
model = KNeighborsClassifier(n_neighbors=5)
```

Train the classifier on the training data

In [44]:
```
model.fit(X_train, y_train)
```

Out[44]:
```
KNeighborsClassifier()
```

Make predictions on the test data

In [45]:
```
y_pred = model.predict(X_test)
```

Print the accuracy

In [46]:
```
KNN_accuracy = model.score(X_test, y_test)
print("K-Nearest Neighbours Accuracy:", KNN_accuracy)
```

```
K-Nearest Neighbours Accuracy: 0.8627548483341622
```

Evaluate the model's performance

In [47]:
```
KNN_report = classification_report(y_test, y_pred)
print(KNN_report)
```

```
              precision    recall  f1-score   support

      Female       0.72      0.82      0.76      2713
        Male       0.93      0.88      0.90      7342

    accuracy                           0.86     10055
   macro avg       0.82      0.85      0.83     10055
weighted avg       0.87      0.86      0.87     10055
```

Test Model on Real Case using Logistic Regression Model

```
In [38]: test_scaled = scaler.transform(test1)
```

```
In [39]: pred = model.predict(test_scaled)
```

```
In [40]: pred
```

```
Out[40]: array(['Male', 'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Female', 'Female', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Female',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male'], dtype=object)
```

```
In [41]: test['gender'].values
```

```
Out[41]: array(['Male', 'Male', 'Male', 'Female', 'Male', 'Female', 'Female',
                'Male', 'Male', 'Female', 'Female', 'Female', 'Female', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Female', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Female', 'Female', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male'], dtype=object)
```

```
In [42]: LG_pred_accuracy = np.mean(pred == test["gender"].values)
         print("Logistic Regression Accuracy:", LG_pred_accuracy)
```

```
Logistic Regression Accuracy: 0.7368421052631579
```

Recognize Gender By using K-Nearest Neighbours (KNN) Model

KNN is a supervised machine learning technique that may be used to solve classification and regression issues. One of the easiest algorithms to learn is K closest neighbour. K closest neighbour is a non-parametric function, i.e. It makes no assumptions about underlying data assumptions. K closest neighbour is also known as a lazy algorithm since it does not learn during the training phase, instead storing the data points and learning during the testing phase.

Create a K-Nearest Neighbours (KNN) Model

```
In [43]: model = KNeighborsClassifier(n_neighbors=5)
```

Train the classifier on the training data

```
In [44]: model.fit(X_train, y_train)
```

```
Out[44]: KNeighborsClassifier()
```

Make predictions on the test data

```
In [45]: y_pred = model.predict(X_test)
```

Print the accuracy

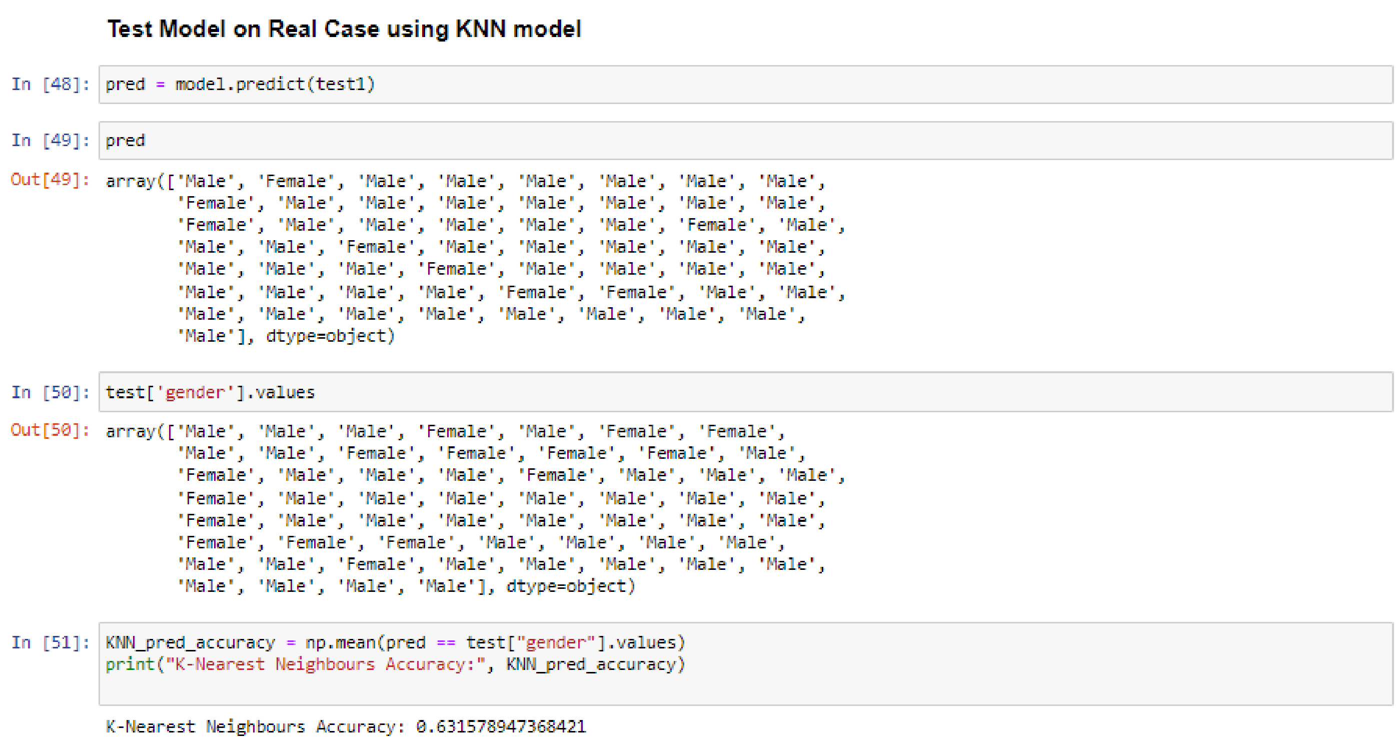
```
In [46]: KNN_accuracy = model.score(X_test, y_test)
         print("K-Nearest Neighbours Accuracy:", KNN_accuracy)
```

```
K-Nearest Neighbours Accuracy: 0.8627548483341622
```

Evaluate the model's performance

```
In [47]: KNN_report = classification_report(y_test, y_pred)
         print(KNN_report)
```

```
              precision    recall  f1-score   support

      Female       0.72      0.82      0.76      2713
        Male       0.93      0.88      0.90      7342

    accuracy                           0.86     10055
   macro avg       0.82      0.85      0.83     10055
weighted avg       0.87      0.86      0.87     10055
```



Test Model on Real Case using KNN model

```
In [48]: pred = model.predict(test1)
```

```
In [49]: pred
```

```
Out[49]: array(['Male', 'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Female', 'Male',
                'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male', 'Female', 'Female', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male'], dtype=object)
```

```
In [50]: test['gender'].values
```

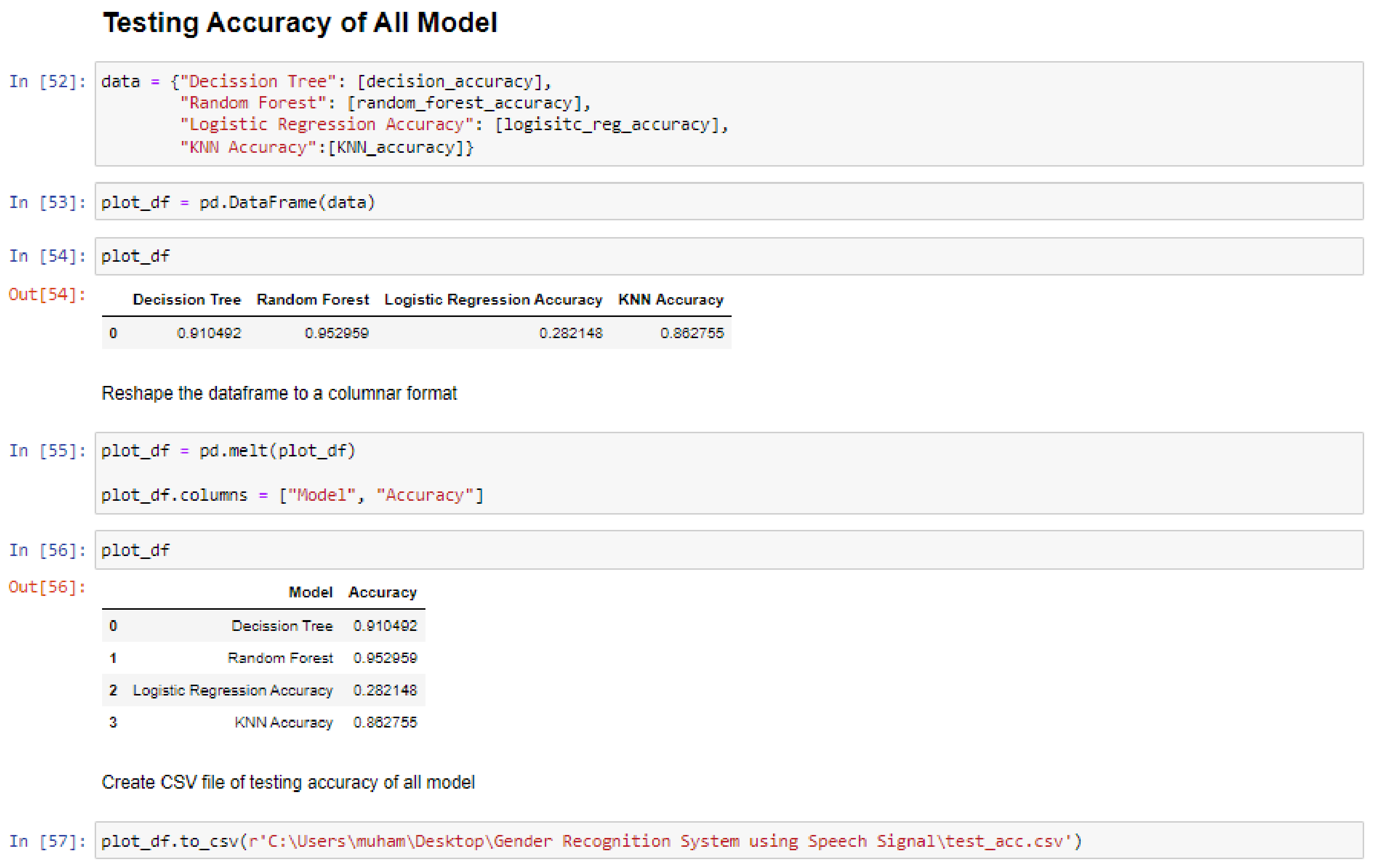
```
Out[50]: array(['Male', 'Male', 'Male', 'Female', 'Male', 'Female', 'Female',
                'Male', 'Male', 'Female', 'Female', 'Female', 'Female', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Female', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Female', 'Female', 'Female', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Male', 'Male',
                'Male', 'Male', 'Male', 'Male'], dtype=object)
```

```
In [51]: KNN_pred_accuracy = np.mean(pred == test["gender"].values)
         print("K-Nearest Neighbours Accuracy:", KNN_pred_accuracy)
```

```
K-Nearest Neighbours Accuracy: 0.631578947368421
```

Data Visualization:

1. Create a data frame with the testing accuracy of all the models.
2. Convert the data frame to a CSV file.



Testing Accuracy of All Model

```
In [52]: data = {"Decission Tree": [decision_accuracy],
                 "Random Forest": [random_forest_accuracy],
                 "Logistic Regression Accuracy": [logisitc_reg_accuracy],
                 "KNN Accuracy":[KNN_accuracy]}
```

```
In [53]: plot_df = pd.DataFrame(data)
```

```
In [54]: plot_df
```

Out[54]:

	Decission Tree	Random Forest	Logistic Regression Accuracy	KNN Accuracy
0	0.910492	0.952959	0.282148	0.862755

Reshape the dataframe to a columnar format

```
In [55]: plot_df = pd.melt(plot_df)

         plot_df.columns = ["Model", "Accuracy"]
```

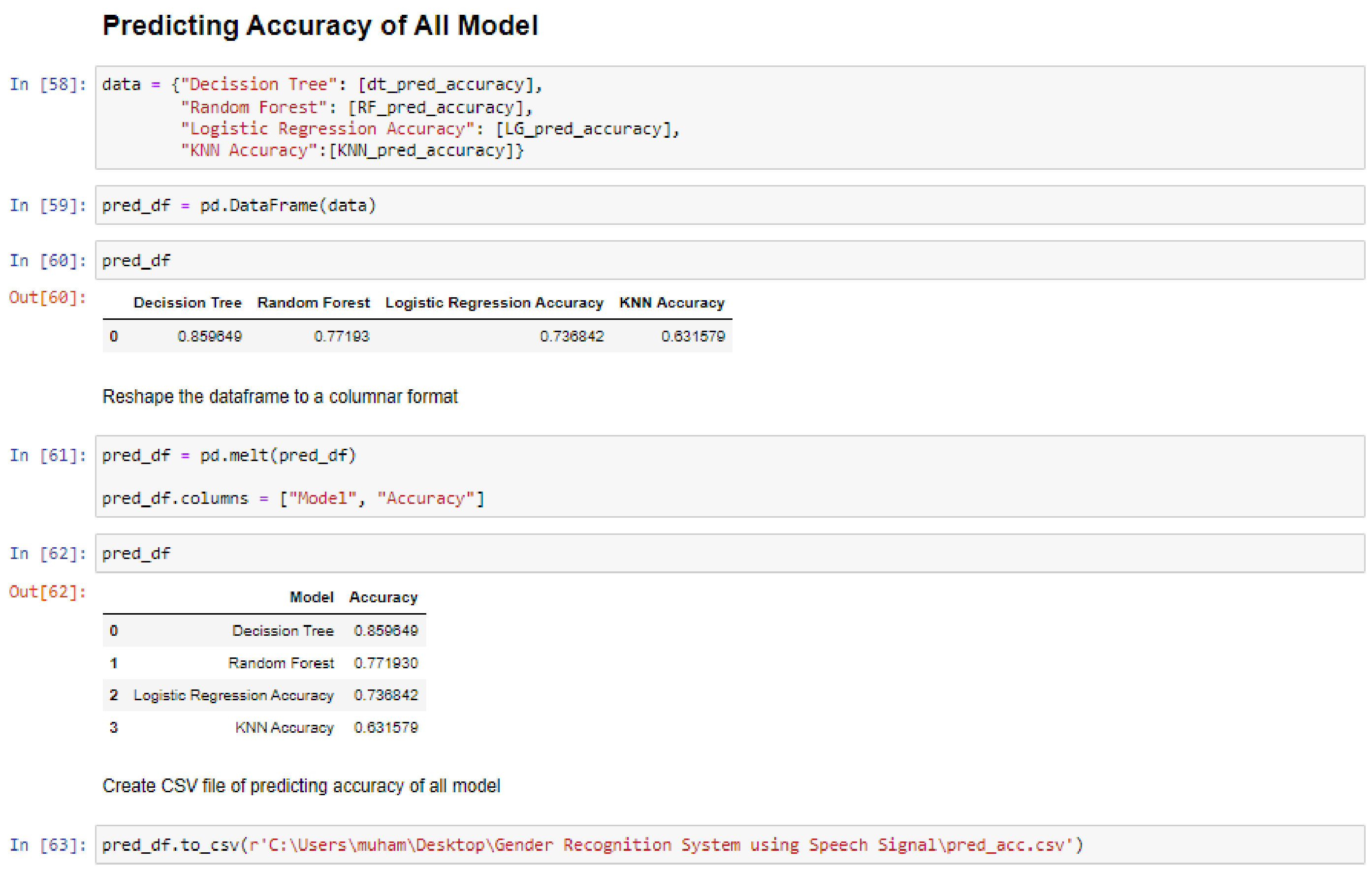
```
In [56]: plot_df
```

Out[56]:

	Model	Accuracy
0	Decission Tree	0.910492
1	Random Forest	0.952959
2	Logistic Regression Accuracy	0.282148
3	KNN Accuracy	0.862755

Create CSV file of testing accuracy of all model

```
In [57]: plot_df.to_csv(r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\test_acc.csv')
```



Predicting Accuracy of All Model

```
In [58]: data = {"Decission Tree": [dt_pred_accuracy],
                "Random Forest": [RF_pred_accuracy],
                "Logistic Regression Accuracy": [LG_pred_accuracy],
                "KNN Accuracy":[KNN_pred_accuracy]}
```

```
In [59]: pred_df = pd.DataFrame(data)
```

```
In [60]: pred_df
```

Out[60]:

	Decission Tree	Random Forest	Logistic Regression Accuracy	KNN Accuracy
0	0.859649	0.77193	0.736842	0.631579

Reshape the dataframe to a columnar format

```
In [61]: pred_df = pd.melt(pred_df)

         pred_df.columns = ["Model", "Accuracy"]
```

```
In [62]: pred_df
```

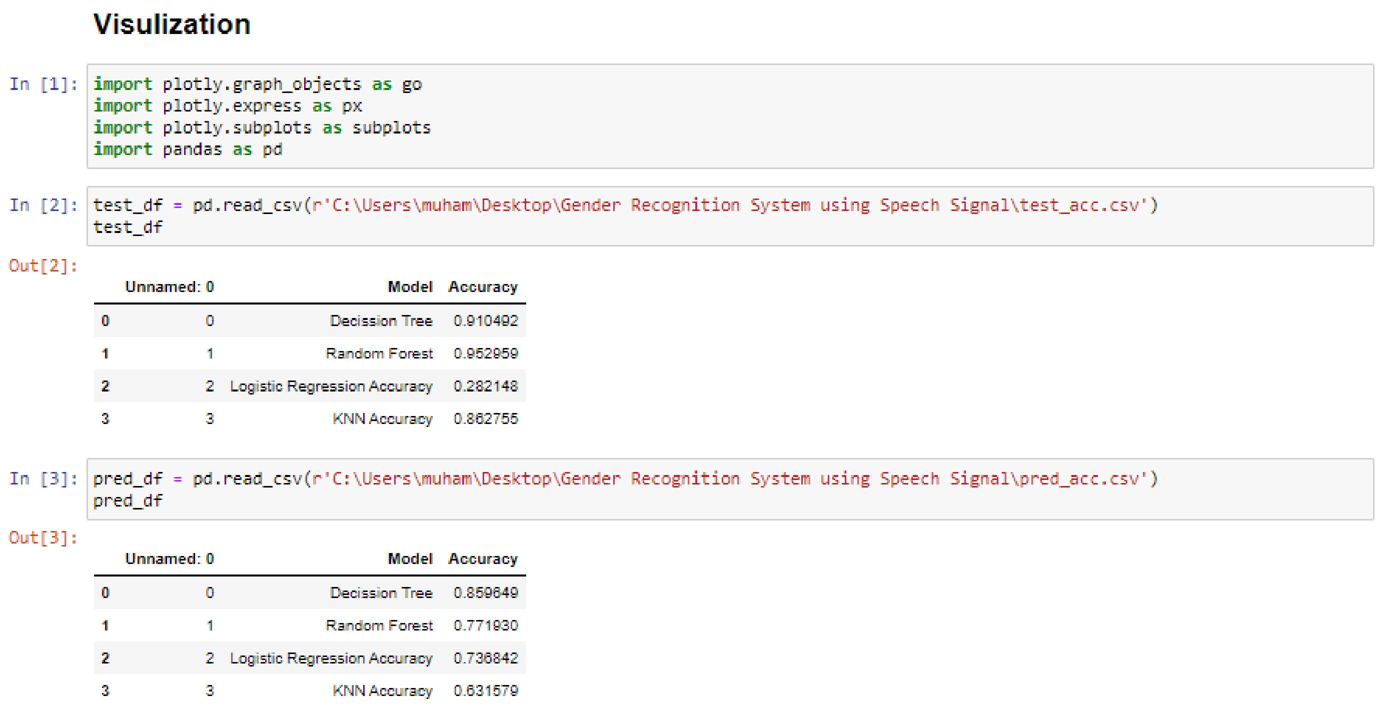
Out[62]:

	Model	Accuracy
0	Decission Tree	0.859649
1	Random Forest	0.771930
2	Logistic Regression Accuracy	0.736842
3	KNN Accuracy	0.631579

Create CSV file of predicting accuracy of all model

```
In [63]: pred_df.to_csv(r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\pred_acc.csv')
```

3. In the final step, the data from the CSV files is read using the Pandas library, and various visualizations such as bar charts, line charts, and scatter plots are created using the Plotly library to compare the testing accuracy and predicting accuracy of the different models.



Visulization

```
In [1]: import plotly.graph_objects as go
        import plotly.express as px
        import plotly.subplots as subplots
        import pandas as pd
```

```
In [2]: test_df = pd.read_csv(r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\test_acc.csv')
        test_df
```

Out[2]:

	Unnamed: 0	Model	Accuracy
0	0	Decission Tree	0.910492
1	1	Random Forest	0.952959
2	2	Logistic Regression Accuracy	0.282148
3	3	KNN Accuracy	0.862755

```
In [3]: pred_df = pd.read_csv(r'C:\Users\muham\Desktop\Gender Recognition System using Speech Signal\pred_acc.csv')
        pred_df
```

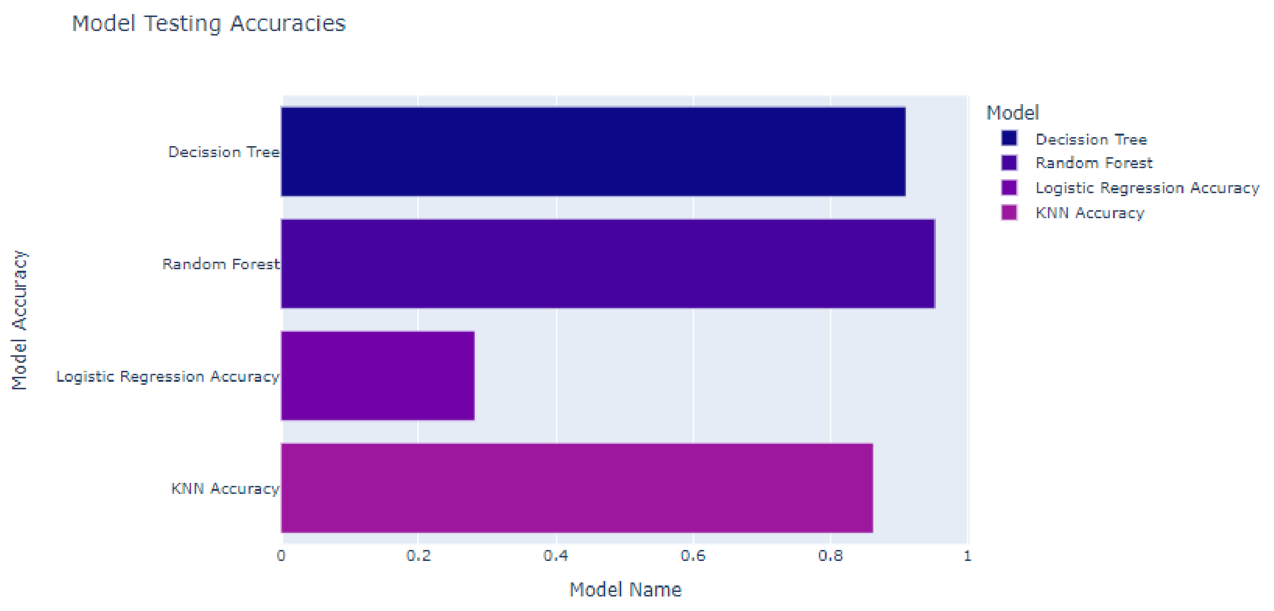
Out[3]:

	Unnamed: 0	Model	Accuracy
0	0	Decission Tree	0.859649
1	1	Random Forest	0.771930
2	2	Logistic Regression Accuracy	0.736842
3	3	KNN Accuracy	0.631579

Visualize Model Testing Accuracy

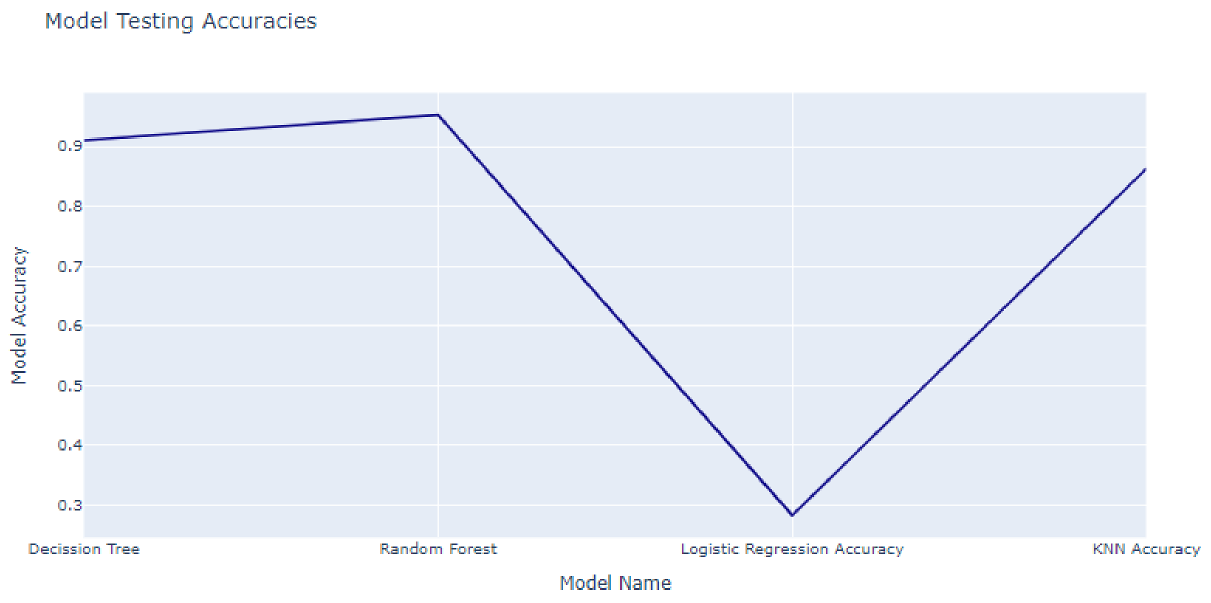
Create the bar chart

In [4]:
```python
fig = px.bar(test_df, x="Accuracy", y="Model", color="Model",
             color_discrete_sequence=px.colors.sequential.Plasma,
             title="Model Testing Accuracies")
# Update the x and y axis labels
fig.update_layout(xaxis_title="Model Name",
                  yaxis_title="Model Accuracy")
# Show the plot
fig.show()
```



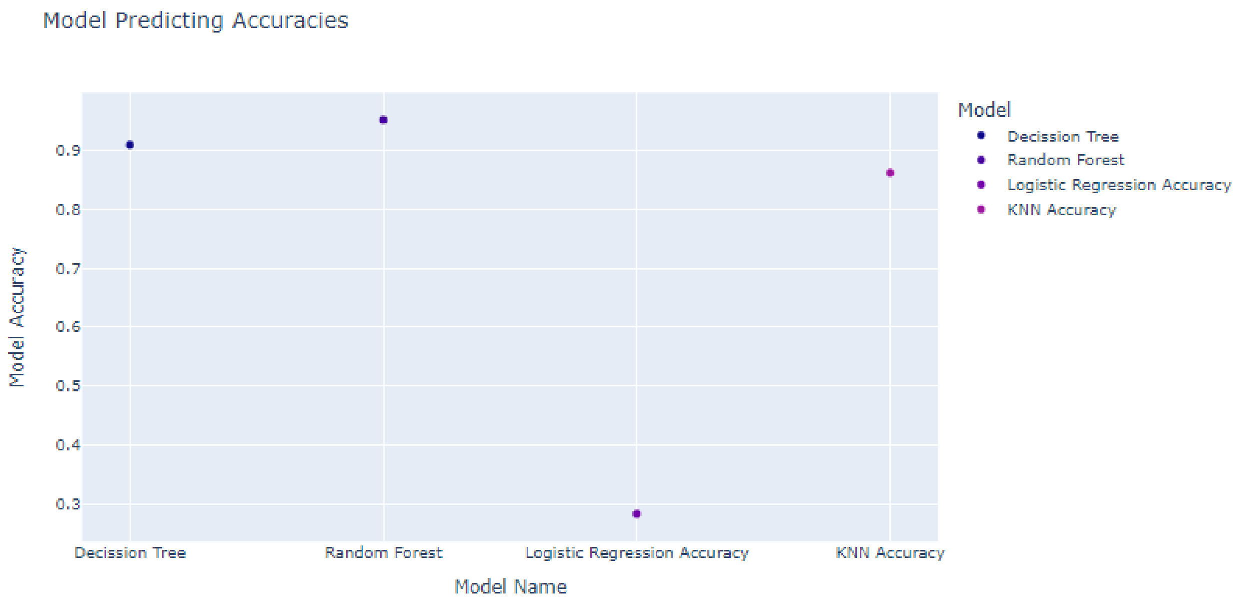
Create the line chart

In [5]:
```python
fig = px.line(test_df, x="Model", y="Accuracy",
             color_discrete_sequence=px.colors.sequential.Plasma,
             title="Model Testing Accuracies")
# Update the x and y axis labels
fig.update_layout(xaxis_title="Model Name",
                  yaxis_title="Model Accuracy")
# Show the plot
fig.show()
```



Create the scatter chart

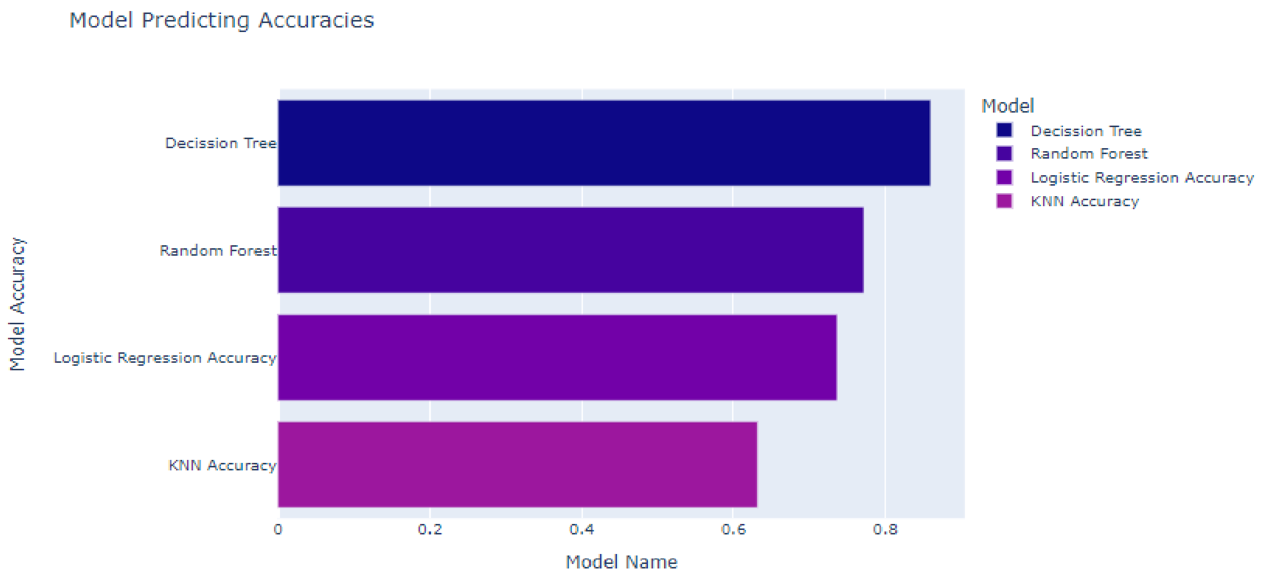
```
In [6]: fig = px.scatter(test_df, x="Model", y="Accuracy", color="Model",
                    color_discrete_sequence=px.colors.sequential.Plasma,
                    title="Model Predicting Accuracies")
        # Update the x and y axis labels
        fig.update_layout(xaxis_title="Model Name",
                          yaxis_title="Model Accuracy")
        # Show the plot
        fig.show()
```



Visualize Model Predicting Accuracy

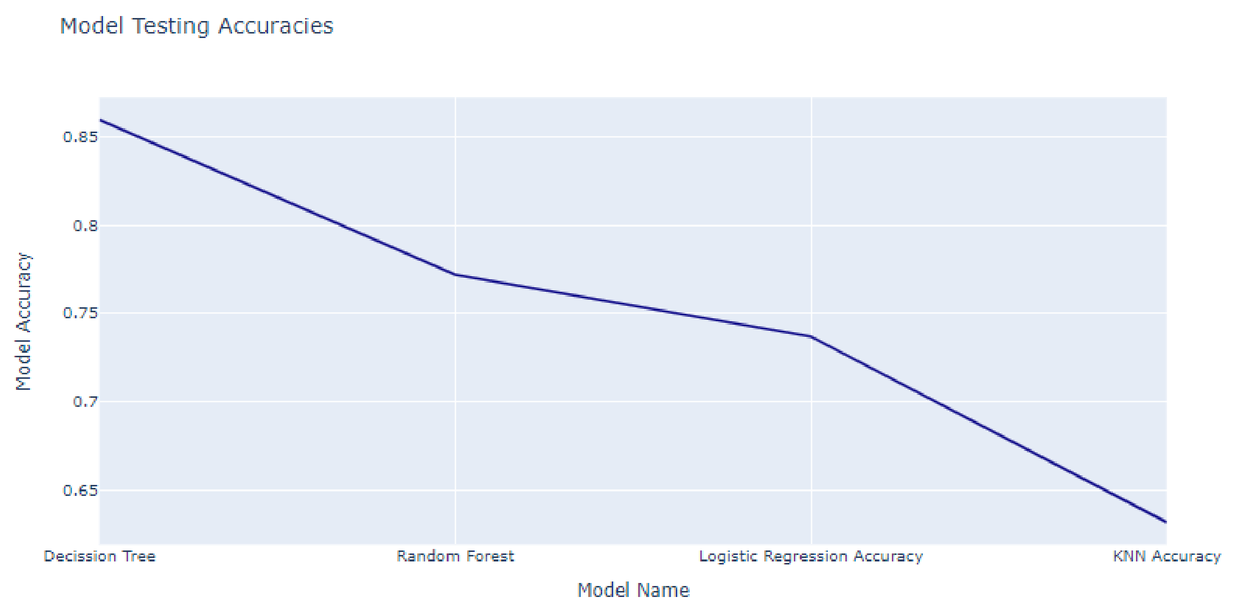
Create the bar chart

```
In [7]: fig = px.bar(pred_df, x="Accuracy", y="Model", color="Model",
                    color_discrete_sequence=px.colors.sequential.Plasma,
                    title="Model Predicting Accuracies")
        # Update the x and y axis labels
        fig.update_layout(xaxis_title="Model Name",
                          yaxis_title="Model Accuracy")
        # Show the plot
        fig.show()
```



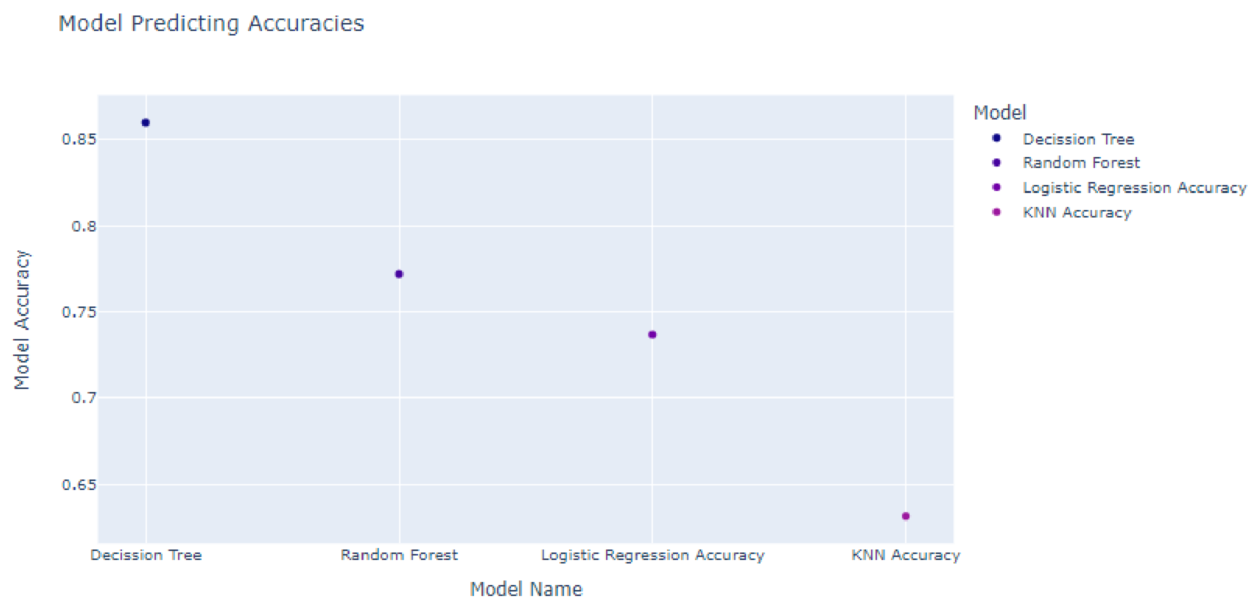
Create the line chart

```
In [8]: fig = px.line(pred_df, x="Model", y="Accuracy",
                     color_discrete_sequence=px.colors.sequential.Plasma,
                     title="Model Testing Accuracies")
        # Update the x and y axis labels
        fig.update_layout(xaxis_title="Model Name",
                          yaxis_title="Model Accuracy")
        # Show the plot
        fig.show()
```



Create the scatter chart

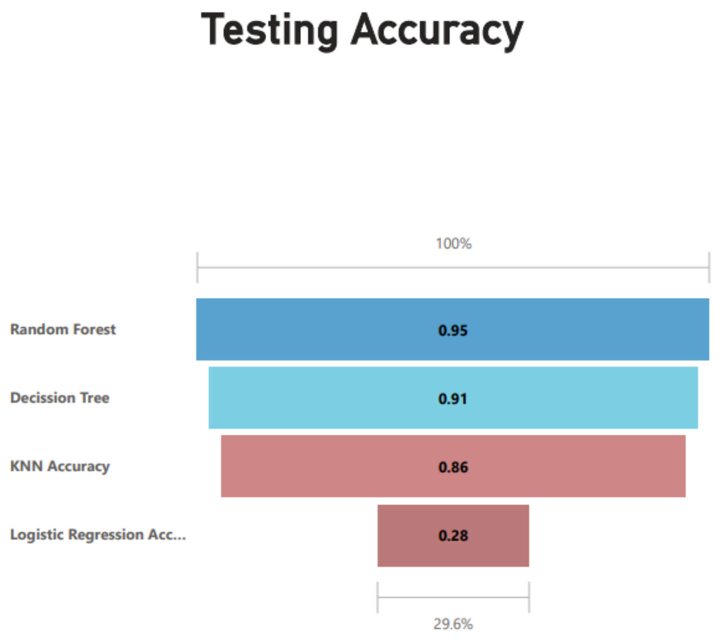
```
In [9]: fig = px.scatter(pred_df, x="Model", y="Accuracy", color="Model",
                     color_discrete_sequence=px.colors.sequential.Plasma,
                     title="Model Predicting Accuracies")
        # Update the x and y axis labels
        fig.update_layout(xaxis_title="Model Name",
                          yaxis_title="Model Accuracy")
        # Show the plot
        fig.show()
```

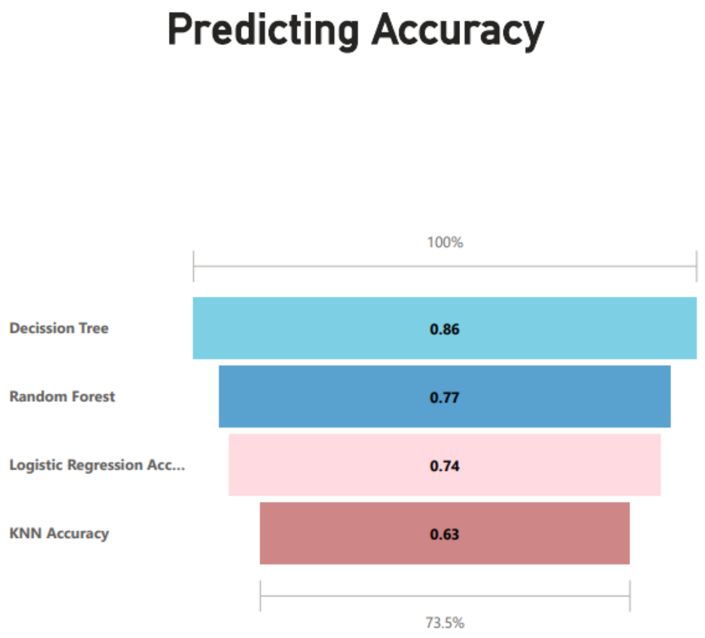


Findings:

The results of our model analysis show that the Random Forest model had the highest testing accuracy at 95.3%, followed by the Decision Tree model with 91.0% accuracy. These models performed significantly better than the Logistic Regression model which had a lower testing accuracy of 28.2%, and the KNN model which had 86.3% accuracy. This suggests that the Random Forest and Decision Tree models are more effective at classifying the gender of individuals based on their speech characteristics.

When it comes to predicting accuracy, the Decision Tree model still performed the best with 85.9% accuracy, while the Random Forest model had a lower accuracy of 77.2%. The Logistic Regression and KNN models had even lower predicting accuracies of 73.7% and 63.2%, respectively. This indicates that the Decision Tree model may be better at generalizing to new data and making accurate predictions.
Overall, the Random Forest and Decision Tree models performed the best in both testing and predicting accuracy. It is important to note that these results should be considered in the context of the specific use case and requirements of the application. Additionally, further tuning and optimization of the models may improve their performance. As a next step, it's worth exploring more complex models such as Neural Networks, Convolutional Neural Networks, or Recurrent Neural Networks.





Conclusion:

In conclusion, our analysis aimed to classify the gender of individuals based on their speech characteristics using various machine learning models. The data used in this analysis consisted of audio files that were extracted from a compressed tar archive and were labeled with the gender of the individual speaking. Features such as the Mel-frequency cepstral coefficients (MFCCs) and pitch of the signal were extracted from the audio files and used as inputs for the models.

The results of the analysis showed that the Random Forest and Decision Tree models performed the best in terms of testing accuracy, with scores of 95.3% and 91.0% respectively. These models outperformed the Logistic Regression and KNN models, which had lower testing accuracy scores of 28.2% and 86.3%. In terms of predicting accuracy, the Decision Tree model had the highest accuracy at 85.9%, followed by the Random Forest model with 77.2%. The Logistic Regression and KNN models had lower predicting accuracies of 73.7% and 63.2% respectively.

It's worth noting that these results should be considered in the context of the specific use case and requirements of the application. Additionally, further tuning and optimization of the models may improve their performance. Further steps could include exploring more complex models such as Neural Networks, Convolutional Neural Networks, or Recurrent Neural Networks. Furthermore, including more data samples and different languages, as well as fine-tuning feature extraction, can increase the accuracy and robustness of the models.

It is also important to consider the ethical and societal implications of using speech characteristics to classify gender. There is a risk of perpetuating harmful stereotypes and biases, and it is important to consider these issues and take steps to mitigate them in any application of this technology. Additionally, it is important to obtain informed consent from individuals before collecting and using their speech data.

In summary, our analysis showed that the Random Forest and Decision Tree models performed the best in classifying the gender of individuals based on their speech characteristics, but it is important to consider the specific use case and ethical implications of the application. Further steps include exploring more complex models, increasing the data sample size and fine-tuning feature extraction.

REFERENCES:

1. Aayush Krishna, Dr. Neelu Jain "Speaker Recognition and Gender Identification using Artificial Neural Network and Support Vector Machine", International Journal for Research in Applied Science & Engineering Technology (IJRASET)ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177Volume 7 Issue VII, July 2019- Available at www.ijraset.com
2. Fadwa Abakarim & Abdenbi Abenaou "Voice Gender Recognition Using Acoustic Features, MFCCs and SVM", International Conference on Computational Science and Its ApplicationsICCSA 2022: Computational Science and Its Applications – ICCSA 2022 pp 634–648
3. "Speaker identification based on MFCC and linear discriminant analysis" by H. Soltani and M. H. Moradi, published in the International Journal of Speech Technology in 2008.
4. "Speaker Gender Identification using Pitch and Formants" by R. R. K. Rao and K. S. R. Anjaneyulu, published in the International Journal of Computer Applications in 2010.