BHAAV VAACHAK

Aditya Agarwal

Abhijeet

Abhilash Sharma

Moradabad Institute of Technology, Moradabad (B.Tech, CSE 2024)

3rd year, session 2022-23

Datasets used in this project

- Crowd-sourced Emotional Mutimodal Actors Dataset (Crema-D)
- Ryerson Audio-Visual Database of Emotional Speech and Song (Ravdess)
- Surrey Audio-Visual Expressed Emotion (Savee)
- Toronto emotional speech set (Tess)

In [1]:
```
import pandas as pd
import numpy as np

import os
import sys

# librosa is a Python library for analyzing audio and music. It can be used to extract the data from
import librosa
import librosa.display
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split

# to play the audio files
from IPython.display import Audio

import keras
from keras.callbacks import ReduceLROnPlateau
from keras.models import Sequential
from keras.layers import Dense, Conv1D, MaxPooling1D, Flatten, Dropout, BatchNormalization
from keras.utils import np_utils, to_categorical
from keras.callbacks import ModelCheckpoint

import warnings
if not sys.warnoptions:
    warnings.simplefilter("ignore")
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

DataSET Loading

In [48]:
```
# Paths for data.
Ravdess = r"C:\Users\adity\Downloads\RAVDESS\audio_speech_actors_01-24"
Crema = r"C:\Users\adity\Downloads\CREMA-D\AudioWAV"
Tess = r"C:\Users\adity\Downloads\TESS\TESS Toronto emotional speech set data\TESS Toronto emotional
Savee = r"C:\Users\adity\Downloads\SAVEE\ALL"
```

1. Ravdess Dataframe

In [55]:
```
ravdess_directory_list = os.listdir(Ravdess)

file_emotion = []
file_path = []
for dir in ravdess_directory_list:
    # as their are 20 different actors in our previous directory we need to extract files for each a
```

```
    actor = os.listdir(Ravdess + '\\'+ dir)
    for file in actor:
        part = file.split('.')[0]
        part = part.split('-')
        # third part in each file represents the emotion associated to that file.
        file_emotion.append(int(part[2]))
        file_path.append(Ravdess +'/'+ dir + '/' + file)

# dataframe for emotion of files
emotion_df = pd.DataFrame(file_emotion, columns=['Emotions'])

# dataframe for path of files.
path_df = pd.DataFrame(file_path, columns=['Path'])
Ravdess_df = pd.concat([emotion_df, path_df], axis=1)

# changing integers to actual emotions.
Ravdess_df.Emotions.replace({1:'neutral', 2:'calm', 3:'happy', 4:'sad', 5:'angry', 6:'fear', 7:'disg
Ravdess_df.head()
```

Out[55]:

	Emotions	Path
0	neutral	C:\Users\adity\Downloads\RAVDESS\audio_speech_...
1	neutral	C:\Users\adity\Downloads\RAVDESS\audio_speech_...
2	neutral	C:\Users\adity\Downloads\RAVDESS\audio_speech_...
3	neutral	C:\Users\adity\Downloads\RAVDESS\audio_speech_...
4	calm	C:\Users\adity\Downloads\RAVDESS\audio_speech_...

2. Crema DataFrame

In [11]:

```
crema_directory_list = os.listdir(Crema)

file_emotion = []
file_path = []

for file in crema_directory_list:
    # storing file paths
    file_path.append(Crema + '\\' + file)
    # storing file emotions
    part=file.split('_')
    if part[2] == 'SAD':
        file_emotion.append('sad')
    elif part[2] == 'ANG':
        file_emotion.append('angry')
    elif part[2] == 'DIS':
        file_emotion.append('disgust')
    elif part[2] == 'FEA':
        file_emotion.append('fear')
    elif part[2] == 'HAP':
        file_emotion.append('happy')
    elif part[2] == 'NEU':
        file_emotion.append('neutral')
    else:
        file_emotion.append('Unknown')

# dataframe for emotion of files
emotion_df = pd.DataFrame(file_emotion, columns=['Emotions'])

# dataframe for path of files.
path_df = pd.DataFrame(file_path, columns=['Path'])
Crema_df = pd.concat([emotion_df, path_df], axis=1)
Crema_df.head()
```

Out[11]:

	Emotions	Path
0	angry	C:\Users\adity\Downloads\CREMA-D\AudioWAV\1001...
1	disgust	C:\Users\adity\Downloads\CREMA-D\AudioWAV\1001...
2	fear	C:\Users\adity\Downloads\CREMA-D\AudioWAV\1001...
3	happy	C:\Users\adity\Downloads\CREMA-D\AudioWAV\1001...
4	neutral	C:\Users\adity\Downloads\CREMA-D\AudioWAV\1001...

3. TESS dataset

In [12]:

```
tess_directory_list = os.listdir(Tess)

file_emotion = []
file_path = []

for dir in tess_directory_list:
    directories = os.listdir(Tess +'\\'+ dir)
    for file in directories:
        part = file.split('.')[0]
        part = part.split('_')[2]
        if part=='ps':
            file_emotion.append('surprise')
        else:
            file_emotion.append(part)
        file_path.append(Tess + dir + '/' + file)

# dataframe for emotion of files
emotion_df = pd.DataFrame(file_emotion, columns=['Emotions'])

# dataframe for path of files.
path_df = pd.DataFrame(file_path, columns=['Path'])
Tess_df = pd.concat([emotion_df, path_df], axis=1)
Tess_df.head()
```

Out[12]:

	Emotions	Path
0	angry	C:\Users\adity\Downloads\TESS\TESS Toronto emo...
1	angry	C:\Users\adity\Downloads\TESS\TESS Toronto emo...
2	angry	C:\Users\adity\Downloads\TESS\TESS Toronto emo...
3	angry	C:\Users\adity\Downloads\TESS\TESS Toronto emo...
4	angry	C:\Users\adity\Downloads\TESS\TESS Toronto emo...

4. CREMA-D dataset

The audio files in this dataset are named in such a way that the prefix letters describes the emotion classes as follows:

- 'a' = 'anger'
- 'd' = 'disgust'
- 'f' = 'fear'
- 'h' = 'happiness'
- 'n' = 'neutral'
- 'sa' = 'sadness'
- 'su' = 'surprise'

In [10]:

```
savee_directory_list = os.listdir(Savee)

file_emotion = []
file_path = []

for file in savee_directory_list:
```

```
    file_path.append(Savee + '\\' + file)
    part = file.split('_')[1]
    ele = part[:-6]
    if ele=='a':
        file_emotion.append('angry')
    elif ele=='d':
        file_emotion.append('disgust')
    elif ele=='f':
        file_emotion.append('fear')
    elif ele=='h':
        file_emotion.append('happy')
    elif ele=='n':
        file_emotion.append('neutral')
    elif ele=='sa':
        file_emotion.append('sad')
    else:
        file_emotion.append('surprise')

# dataframe for emotion of files
emotion_df = pd.DataFrame(file_emotion, columns=['Emotions'])

# dataframe for path of files.
path_df = pd.DataFrame(file_path, columns=['Path'])
Savee_df = pd.concat([emotion_df, path_df], axis=1)
Savee_df.head()
```

Out[10]:

	Emotions	Path
0	angry	C:\Users\adity\Downloads\SAVEE\ALL\DC_a01.wav
1	angry	C:\Users\adity\Downloads\SAVEE\ALL\DC_a02.wav
2	angry	C:\Users\adity\Downloads\SAVEE\ALL\DC_a03.wav
3	angry	C:\Users\adity\Downloads\SAVEE\ALL\DC_a04.wav
4	angry	C:\Users\adity\Downloads\SAVEE\ALL\DC_a05.wav

In [56]:

```
# creating Dataframe using all the 4 dataframes we created so far.
data_path = pd.concat([Ravdess_df, Crema_df, Tess_df, Savee_df], axis = 0)
data_path.to_csv("data_path.csv",index=False)
data_path
```

Out[56]:

	Emotions	Path
0	neutral	C:\Users\adity\Downloads\RAVDESS\audio_speech_...
1	neutral	C:\Users\adity\Downloads\RAVDESS\audio_speech_...
2	neutral	C:\Users\adity\Downloads\RAVDESS\audio_speech_...
3	neutral	C:\Users\adity\Downloads\RAVDESS\audio_speech_...
4	calm	C:\Users\adity\Downloads\RAVDESS\audio_speech_...
...	...	...
475	surprise	C:\Users\adity\Downloads\SAVEE\ALL\KL_su11.wav
476	surprise	C:\Users\adity\Downloads\SAVEE\ALL\KL_su12.wav
477	surprise	C:\Users\adity\Downloads\SAVEE\ALL\KL_su13.wav
478	surprise	C:\Users\adity\Downloads\SAVEE\ALL\KL_su14.wav
479	surprise	C:\Users\adity\Downloads\SAVEE\ALL\KL_su15.wav

12162 rows × 2 columns

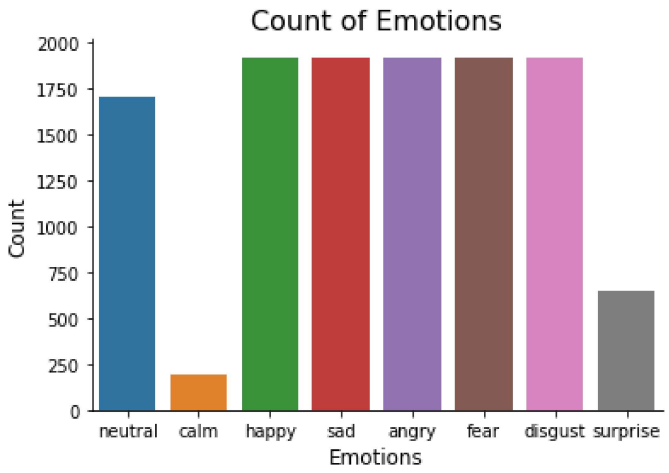
Data Visualisation and Exploration

First let's plot the count of each emotions in our dataset.

In [52]:

```
plt.title('Count of Emotions', size=16)
sns.countplot(data_path.Emotions)
```

```
plt.ylabel('Count', size=12)
plt.xlabel('Emotions', size=12)
sns.despine(top=True, right=True, left=False, bottom=False)
plt.show()
```

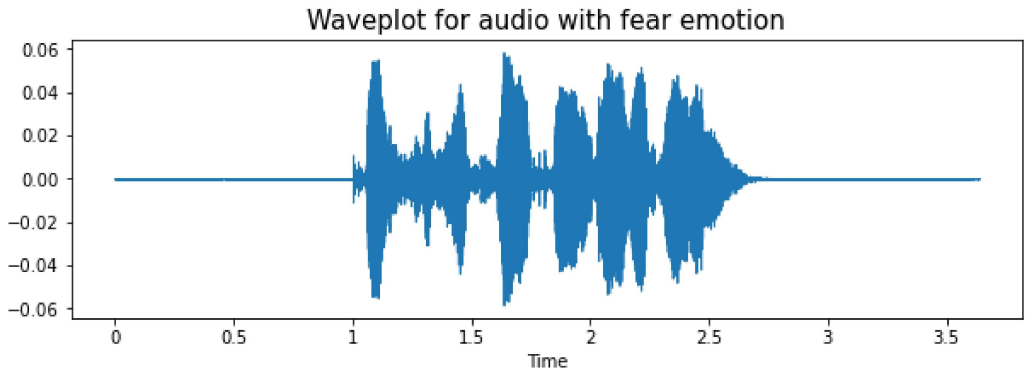


We can also plot waveplots and spectograms for audio signals

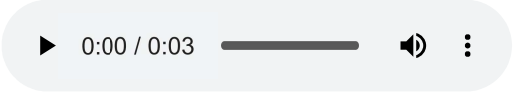
- Waveplots - Waveplots let us know the loudness of the audio at a given time.
- Spectograms - A spectrogram is a visual representation of the spectrum of frequencies of sound or other signals as they vary with time. It's a representation of frequencies changing with respect to time for given audio/music signals.

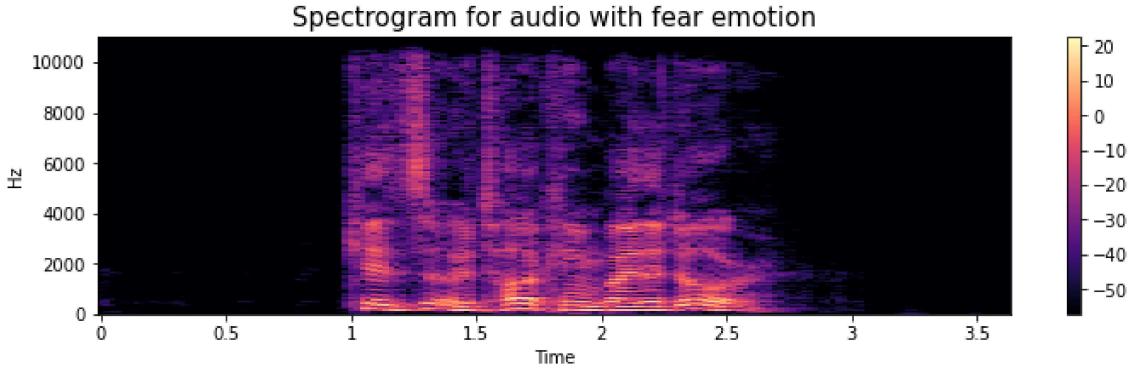
In [57]:
```
def create_waveplot(data, sr, e):
    plt.figure(figsize=(10, 3))
    plt.title('Waveplot for audio with {} emotion'.format(e), size=15)
    librosa.display.waveshow(data, sr=sr)
    plt.show()

def create_spectrogram(data, sr, e):
    # stft function converts the data into short term fourier transform
    X = librosa.stft(data)
    Xdb = librosa.amplitude_to_db(abs(X))
    plt.figure(figsize=(12, 3))
    plt.title('Spectrogram for audio with {} emotion'.format(e), size=15)
    librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
    #librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='log')
    plt.colorbar()
```

In [58]:
```
emotion='fear'
path = np.array(data_path.Path[data_path.Emotions==emotion])[1]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)
```

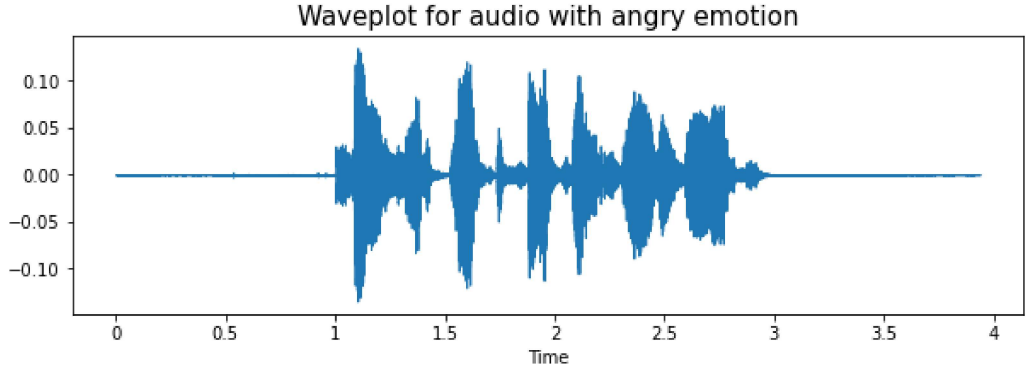


Out[58]:

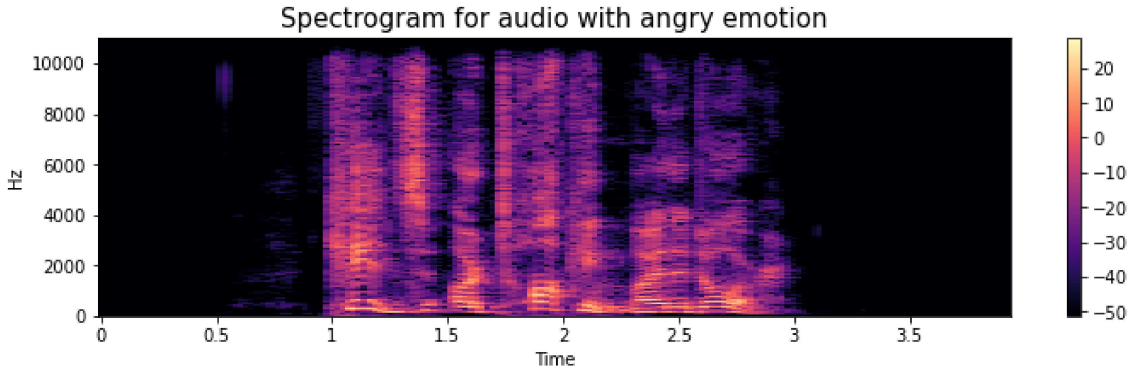




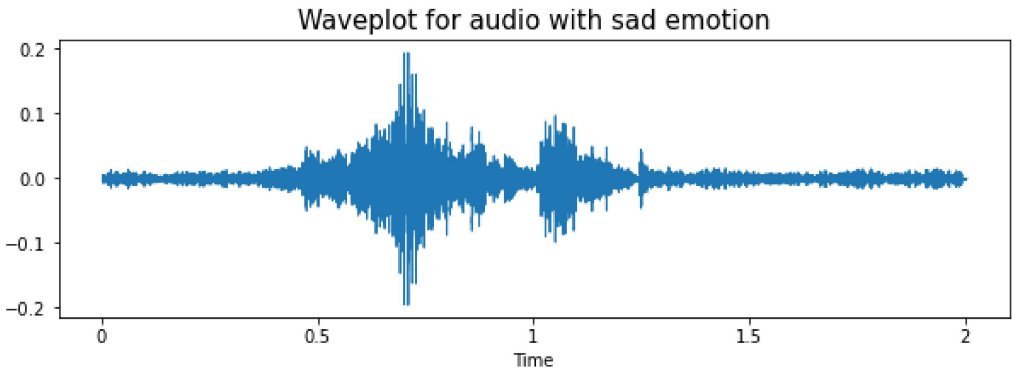
In [59]:
```
emotion='angry'
path = np.array(data_path.Path[data_path.Emotions==emotion])[1]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)
```



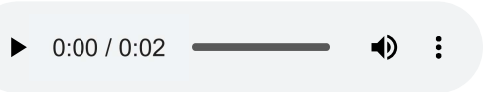
Out[59]:

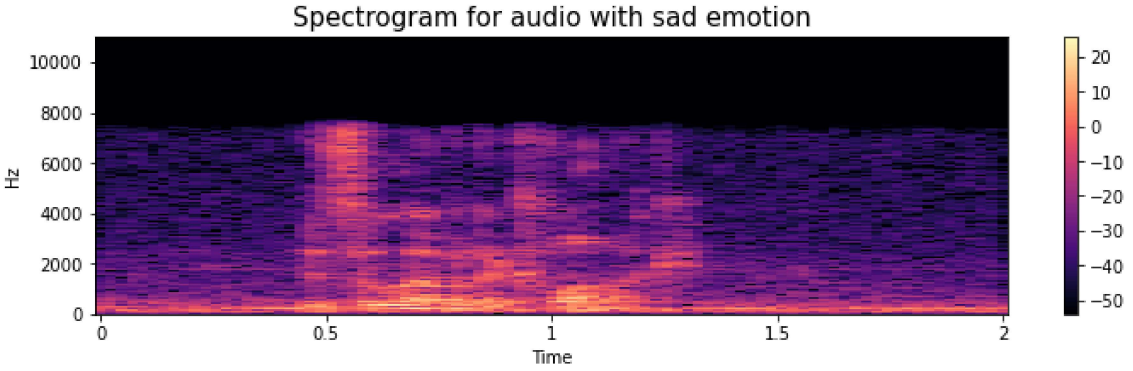


In [20]:
```
emotion='sad'
path = np.array(data_path.Path[data_path.Emotions==emotion])[1]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)
```



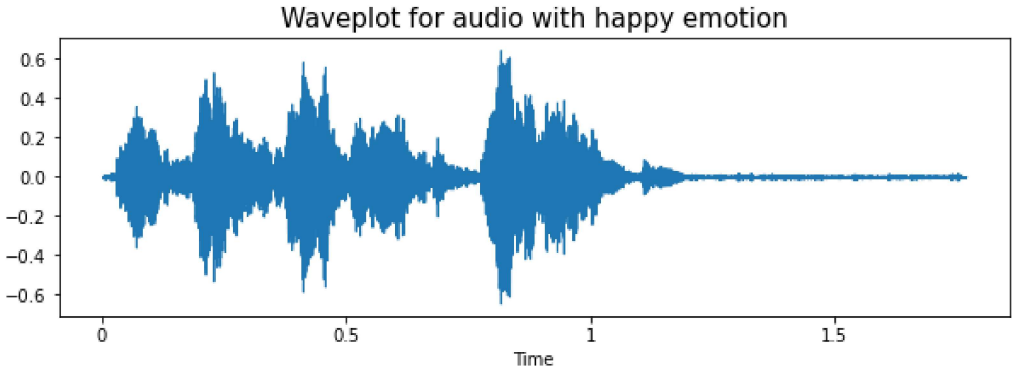
Out[20]:



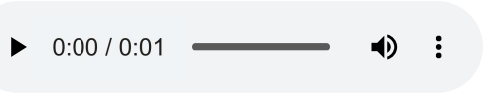


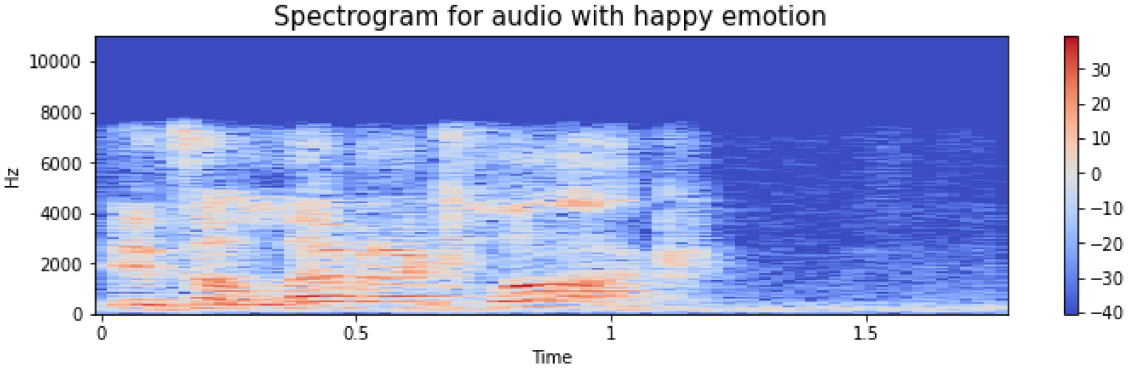
In [21]:

```
emotion='happy'
path = np.array(data_path.Path[data_path.Emotions==emotion])[1]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)
```



Out[21]:





Data Augmentation

- Data augmentation is the process by which we create new synthetic data samples by adding small perturbations on our initial training set.
- To generate syntactic data for audio, we can apply noise injection, shifting time, changing pitch and speed.
- The objective is to make our model invariant to those perturbations and enhace its ability to generalize.
- In order to this to work adding the perturbations must conserve the same label as the original training sample.
- In images data augmention can be performed by shifting the image, zooming, rotating ...

First, let's check which augmentation techniques works better for our dataset.

In [22]:
```
def noise(data):
    noise_amp = 0.035*np.random.uniform()*np.amax(data)
    data = data + noise_amp*np.random.normal(size=data.shape[0])
    return data

def stretch(data, rate=0.8):
    return librosa.effects.time_stretch(data, rate)

def shift(data):
    shift_range = int(np.random.uniform(low=-5, high = 5)*1000)
    return np.roll(data, shift_range)

def pitch(data, sampling_rate, pitch_factor=0.7):
    return librosa.effects.pitch_shift(data, sampling_rate, pitch_factor)

# taking any example and checking for techniques.
path = np.array(data_path.Path)[1]
data, sample_rate = librosa.load(path)
```

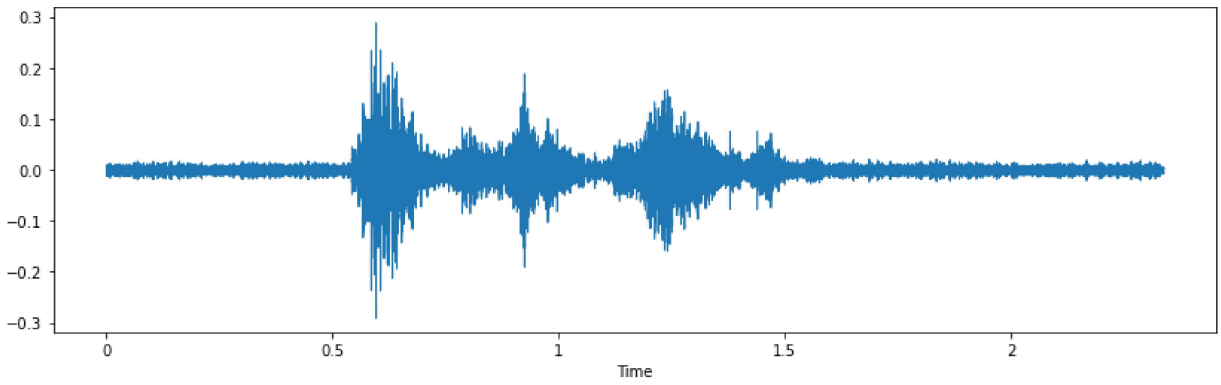
1. Simple Audio

In [ ]:
```
plt.figure(figsize=(14,4))
librosa.display.waveshow(y=data, sr=sample_rate)
Audio(path)
```

2. Noise Injection

In [25]:
```
x = noise(data)
plt.figure(figsize=(14,4))
librosa.display.waveshow(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

Out[25]:



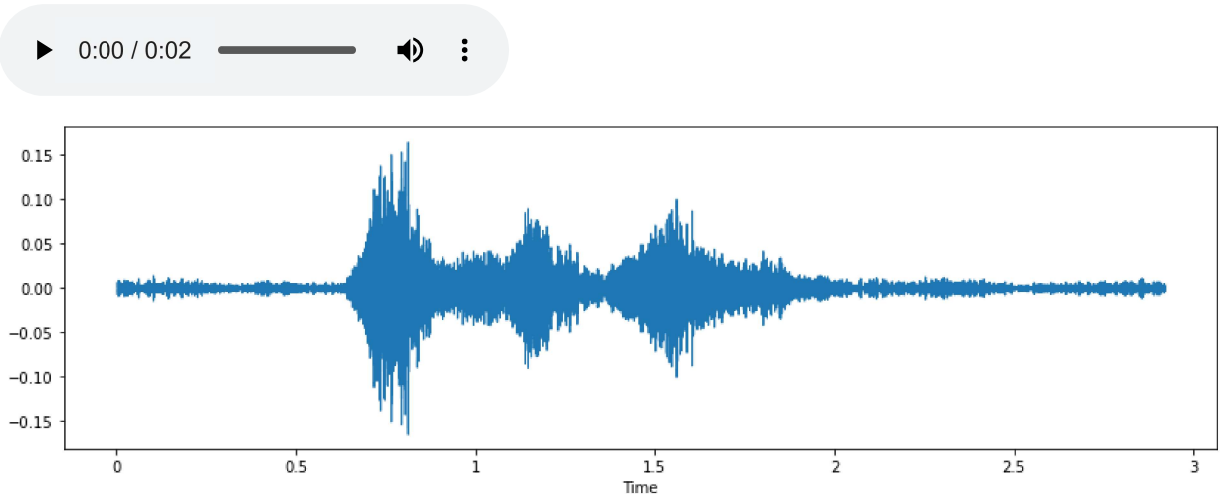
We can see noise injection is a very good augmentation technique because of which we can assure our training model is not overfitted

3. Stretching

In [27]:
```
x = stretch(data)
plt.figure(figsize=(14,4))
```

```
librosa.display.waveshow(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

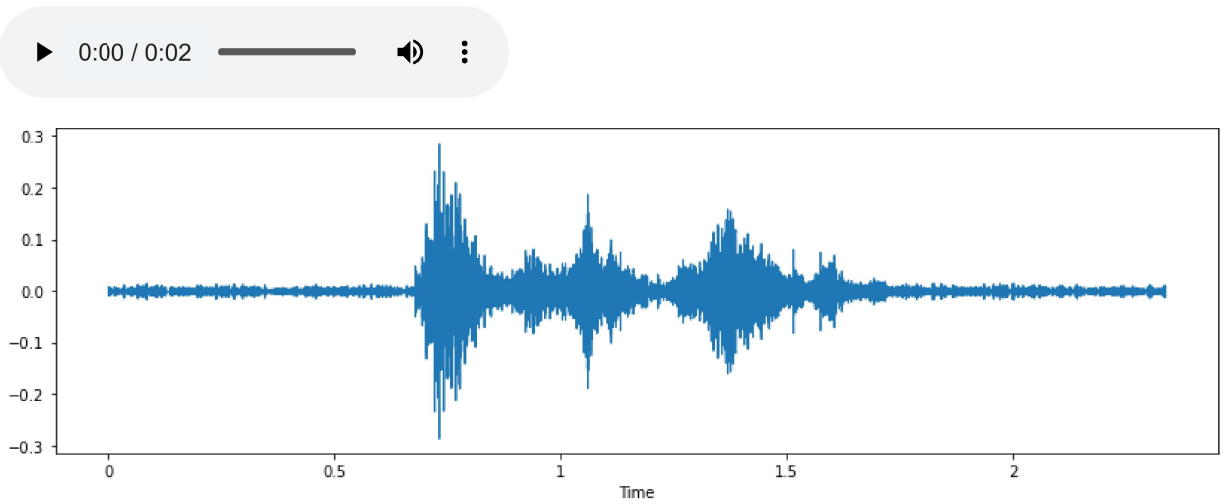
Out[27]:



4. Shifting

In [28]:
```
x = shift(data)
plt.figure(figsize=(14,4))
librosa.display.waveshow(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

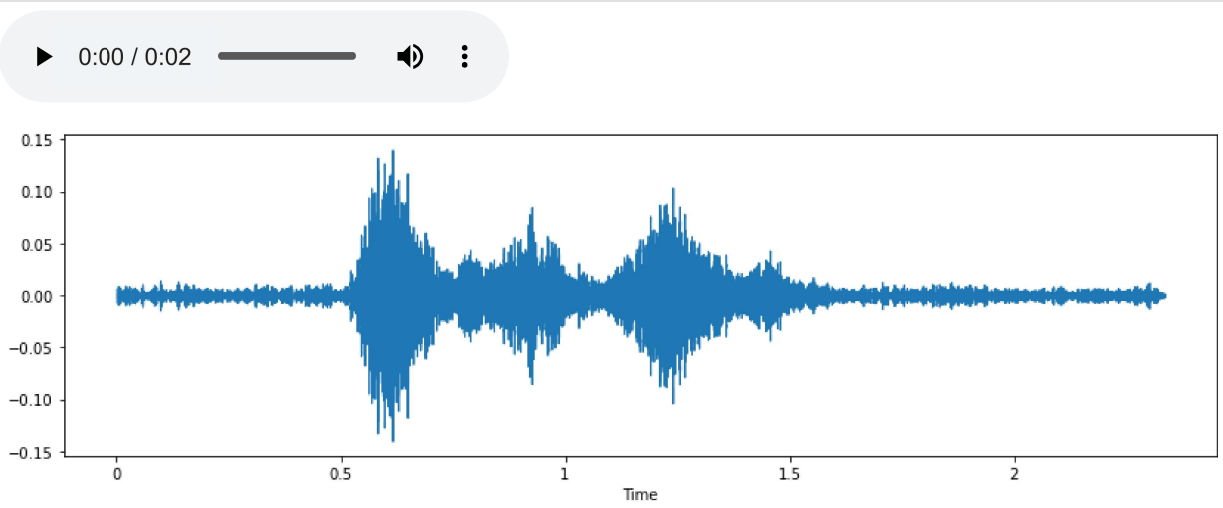
Out[28]:



5. Pitch

In [29]:
```
x = pitch(data, sample_rate)
plt.figure(figsize=(14,4))
librosa.display.waveshow(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

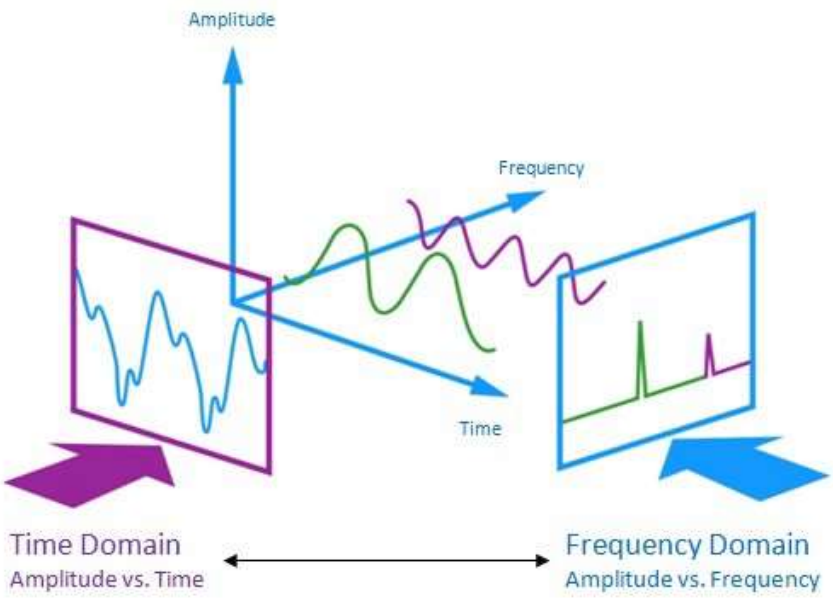
Out[29]:



- From the above types of augmentation techniques i am using noise, stretching(ie. changing speed) and some pitching.

Feature Extraction

The audio signal is a three-dimensional signal in which three axes represent time, amplitude and frequency.



- Zero Crossing Rate
- Chroma_stft
- MFCC
- RMS(root mean square) value
- MelSpectogram to train our model.

In [30]:
```
def extract_features(data):
    # ZCR
    result = np.array([])
    zcr = np.mean(librosa.feature.zero_crossing_rate(y=data).T, axis=0)
    result=np.hstack((result, zcr)) # stacking horizontally

    # Chroma_stft
    stft = np.abs(librosa.stft(data))
    chroma_stft = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T, axis=0)
    result = np.hstack((result, chroma_stft)) # stacking horizontally

    # MFCC
    mfcc = np.mean(librosa.feature.mfcc(y=data, sr=sample_rate).T, axis=0)
    result = np.hstack((result, mfcc)) # stacking horizontally

    # Root Mean Square Value
    rms = np.mean(librosa.feature.rms(y=data).T, axis=0)
    result = np.hstack((result, rms)) # stacking horizontally

    # MelSpectogram
    mel = np.mean(librosa.feature.melspectrogram(y=data, sr=sample_rate).T, axis=0)
    result = np.hstack((result, mel)) # stacking horizontally

    return result

def get_features(path):
    # duration and offset are used to take care of the no audio in start and the ending of each audi
    data, sample_rate = librosa.load(path, duration=2.5, offset=0.6)

    # without augmentation
    res1 = extract_features(data)
    result = np.array(res1)

    # data with noise
```

```
    noise_data = noise(data)
    res2 = extract_features(noise_data)
    result = np.vstack((result, res2)) # stacking vertically

    # data with stretching and pitching
    new_data = stretch(data)
    data_stretch_pitch = pitch(new_data, sample_rate)
    res3 = extract_features(data_stretch_pitch)
    result = np.vstack((result, res3)) # stacking vertically

    return result
```

In [31]:

```
X, Y = [], []
for path, emotion in zip(data_path.Path, data_path.Emotions):
    feature = get_features(path)
    for ele in feature:
        X.append(ele)
        # appending emotion 3 times as we have made 3 augmentation techniques on each audio file.
        Y.append(emotion)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Input In [31], in <cell line: 2>()
      1 X, Y = [], []
      2 for path, emotion in zip(data_path.Path, data_path.Emotions):
----> 3     feature = get_features(path)
      4     for ele in feature:
      5         X.append(ele)

Input In [30], in get_features(path)
     39 # data with stretching and pitching
     40 new_data = stretch(data)
---> 41 data_stretch_pitch = pitch(new_data, sample_rate)
     42 res3 = extract_features(data_stretch_pitch)
     43 result = np.vstack((result, res3)) # stacking vertically

Input In [22], in pitch(data, sampling_rate, pitch_factor)
     13 def pitch(data, sampling_rate, pitch_factor=0.7):
---> 14     return librosa.effects.pitch_shift(data, sampling_rate, pitch_factor)

File ~\anaconda3\lib\site-packages\librosa\util\decorators.py:104, in deprecate_positional_args.<locals>._inner_deprecate_positional_args.<locals>.inner_f(*args, **kwargs)
     96 warnings.warn(
     97     f"Pass {args_msg} as keyword args. From version "
     98     f"{version} passing these as positional arguments "
   (...)
    101     stacklevel=2,
    102 )
    103 kwargs.update(zip(sig.parameters, args))
--> 104 return f(**kwargs)

File ~\anaconda3\lib\site-packages\librosa\effects.py:327, in pitch_shift(y, sr, n_steps, bins_per_octave, res_type, **kwargs)
    324 rate = 2.0 ** (-float(n_steps) / bins_per_octave)
    326 # Stretch in time, then resample
--> 327 y_shift = core.resample(
    328     time_stretch(y, rate=rate, **kwargs),
    329     orig_sr=float(sr) / rate,
    330     target_sr=sr,
    331     res_type=res_type,
    332 )
    334 # Crop to the same dimension as the input
    335 return util.fix_length(y_shift, size=y.shape[-1])

File ~\anaconda3\lib\site-packages\librosa\util\decorators.py:88, in deprecate_positional_args.<locals>._inner_deprecate_positional_args.<locals>.inner_f(*args, **kwargs)
     86 extra_args = len(args) - len(all_args)
     87 if extra_args <= 0:
---> 88     return f(*args, **kwargs)
     90 # extra_args > 0
     91 args_msg = [
     92     "{}={}".format(name, arg)
     93     for name, arg in zip(kwonly_args[:extra_args], args[-extra_args:])
     94 ]

File ~\anaconda3\lib\site-packages\librosa\core\audio.py:647, in resample(y, orig_sr, target_sr, res_type, fix, scale, **kwargs)
    645     y_hat = soxr.resample(y.T, orig_sr, target_sr, quality=res_type).T
    646 else:
--> 647     y_hat = resampy.resample(y, orig_sr, target_sr, filter=res_type, axis=-1)
    649 if fix:
    650     y_hat = util.fix_length(y_hat, size=n_samples, **kwargs)

File ~\anaconda3\lib\site-packages\resampy\core.py:168, in resample(x, sr_orig, sr_new, axis, filter, parallel, **kwargs)
    158         resample_f_s(
    159             x.swapaxes(-1, axis),
    160             t_out,
   (...)
    165             y.swapaxes(-1, axis),
    166         )
    167 else:
--> 168     resample_f_s(
    169         x.swapaxes(-1, axis),
    170         t_out,
```

```
    171         interp win,
    172         interp_delta,
    173         precision,
    174         scale,
    175         y.swapaxes(-1, axis),
    176     )
    178 return y

File ~\anaconda3\lib\site-packages\numba\np\ufunc\gufunc.py:192, in GUFunc.__call__(self, *args, **k
wargs)
    190     self.add(sig)
    191     self.build_ufunc()
--> 192 return self.ufunc(*args, **kwargs)

KeyboardInterrupt:
```

In [32]:
```
len(X), len(Y), data_path.Path.shape
```

Out[32]:
```
(4545, 4545, (10722,))
```

In [47]:
```
Features = pd.DataFrame(X)
Features['labels'] = Y
Features.to_csv('features.csv', index=False)
Features.head()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3799, in DataFrame._set_item_mgr(self, key,
 value)
   3798 try:
-> 3799     loc = self._info_axis.get_loc(key)
   3800 except KeyError:
   3801     # This item wasn't present, just insert at end

File ~\anaconda3\lib\site-packages\pandas\core\indexes\range.py:389, in RangeIndex.get_loc(self, ke
y, method, tolerance)
    388     self._check_indexing_error(key)
--> 389     raise KeyError(key)
    390 return super().get_loc(key, method=method, tolerance=tolerance)

KeyError: 'labels'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
Input In [47], in <cell line: 2>()
      1 Features = pd.DataFrame(X)
----> 2 Features['labels'] = Y
      3 Features.to_csv('features.csv', index=False)
      4 Features.head()

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3655, in DataFrame.__setitem__(self, key, va
lue)
   3652     self._setitem_array([key], value)
   3653 else:
   3654     # set column
-> 3655     self._set_item(key, value)

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3845, in DataFrame._set_item(self, key, valu
e)
   3842         if isinstance(existing_piece, DataFrame):
   3843             value = np.tile(value, (len(existing_piece.columns), 1)).T
-> 3845 self._set_item_mgr(key, value)

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3802, in DataFrame._set_item_mgr(self, key,
 value)
   3799     loc = self._info_axis.get_loc(key)
   3800 except KeyError:
   3801     # This item wasn't present, just insert at end
-> 3802     self._mgr.insert(len(self._info_axis), key, value)
   3803 else:
   3804     self._iset_item_mgr(loc, value)

File ~\anaconda3\lib\site-packages\pandas\core\internals\managers.py:1235, in BlockManager.insert(se
lf, loc, item, value)
   1233     value = value.T
   1234     if len(value) > 1:
-> 1235         raise ValueError(
   1236             f"Expected a 1D array, got an array with shape {value.T.shape}"
   1237         )
   1238 else:
   1239     value = ensure_block_shape(value, ndim=self.ndim)

ValueError: Expected a 1D array, got an array with shape (4545, 6)
```

- We have applied data augmentation and extracted the features for each audio files and saved them.

Data Preparation

- As of now we have extracted the data, now we need to normalize and split our data for training and testing.

In [34]:
```
X = Features.iloc[: ,:-1].values
Y = Features['labels'].values
```

In [35]:
```
# As this is a multiclass classification problem onehotencoding our Y.
encoder = OneHotEncoder()
```

```
Y = encoder.fit_transform(np.array(Y).reshape(-1,1)).toarray()
```

In [36]:
```
# splitting data
x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state=0, shuffle=True)
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[36]:
```
((3408, 162), (3408, 6), (1137, 162), (1137, 6))
```

In [37]:
```
# scaling our data with sklearn's Standard scaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[37]:
```
((3408, 162), (3408, 6), (1137, 162), (1137, 6))
```

In [38]:
```
# making our data compatible to model.
x_train = np.expand_dims(x_train, axis=2)
x_test = np.expand_dims(x_test, axis=2)
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[38]:
```
((3408, 162, 1), (3408, 6), (1137, 162, 1), (1137, 6))
```

Modelling

In [39]:
```
model=Sequential()
model.add(Conv1D(256, kernel_size=5, strides=1, padding='same', activation='relu', input_shape=(x_tr
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))

model.add(Conv1D(256, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))

model.add(Conv1D(128, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
model.add(Dropout(0.2))

model.add(Conv1D(64, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))

model.add(Flatten())
model.add(Dense(units=32, activation='relu'))
model.add(Dropout(0.3))

model.add(Dense(units=8, activation='softmax'))
model.compile(optimizer = 'adam' , loss = 'categorical_crossentropy' , metrics = ['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 162, 256)          1536

 max_pooling1d (MaxPooling1D  (None, 81, 256)          0
 )

 conv1d_1 (Conv1D)           (None, 81, 256)           327936

 max_pooling1d_1 (MaxPooling  (None, 41, 256)          0
 1D)

 conv1d_2 (Conv1D)           (None, 41, 128)           163968

 max_pooling1d_2 (MaxPooling  (None, 21, 128)          0
 1D)

 dropout (Dropout)           (None, 21, 128)           0

 conv1d_3 (Conv1D)           (None, 21, 64)            41024

 max_pooling1d_3 (MaxPooling  (None, 11, 64)           0
 1D)

 flatten (Flatten)           (None, 704)               0

 dense (Dense)               (None, 32)                22560

 dropout_1 (Dropout)         (None, 32)                0

 dense_1 (Dense)             (None, 8)                 264

=================================================================
Total params: 557,288
Trainable params: 557,288
Non-trainable params: 0
_________________________________________________________________
```

In [40]:
```
rlrp = ReduceLROnPlateau(monitor='loss', factor=0.4, verbose=0, patience=2, min_lr=0.0000001)
history=model.fit(x_train, y_train, batch_size=64, epochs=50, validation_data=(x_test, y_test), call
```

```
Epoch 1/50
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [40], in <cell line: 2>()
      1 rlrp = ReduceLROnPlateau(monitor='loss', factor=0.4, verbose=0, patience=2, min_lr=0.0000001
)
----> 2 history=model.fit(x_train, y_train, batch_size=64, epochs=50, validation_data=(x_test, y_tes
t), callbacks=[rlrp])

File ~\anaconda3\lib\site-packages\keras\utils\traceback_utils.py:70, in filter_traceback.<locals>.e
rror_handler(*args, **kwargs)
     67     filtered_tb = _process_traceback_frames(e.__traceback__)
     68     # To get the full stack trace, call:
     69     # `tf.debugging.disable_traceback_filtering()`
---> 70     raise e.with_traceback(filtered_tb) from None
     71 finally:
     72     del filtered_tb

File ~\AppData\Local\Temp\__autograph_generated_filegzaidau2.py:15, in outer_factory.<locals>.inner_
factory.<locals>.tf__train_function(iterator)
     13 try:
     14     do_return = True
---> 15     retval_ = ag__.converted_call(ag__.ld(step_function), (ag__.ld(self), ag__.ld(iterato
r)), None, fscope)
     16 except:
     17     do_return = False

ValueError: in user code:

    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\training.py", line 1160, in train_
function  *
        return step_function(self, iterator)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\training.py", line 1146, in step_f
unction  **
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\training.py", line 1135, in run_st
ep  **
        outputs = model.train_step(data)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\training.py", line 994, in train_s
tep
        loss = self.compute_loss(x, y, y_pred, sample_weight)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\training.py", line 1052, in comput
e_loss
        return self.compiled_loss(
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\compile_utils.py", line 265, in __
call__
        loss_value = loss_obj(y_t, y_p, sample_weight=sw)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\losses.py", line 152, in __call__
        losses = call_fn(y_true, y_pred)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\losses.py", line 272, in call  **
        return ag_fn(y_true, y_pred, **self._fn_kwargs)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\losses.py", line 1990, in categorical_cro
ssentropy
        return backend.categorical_crossentropy(
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\backend.py", line 5529, in categorical_cr
ossentropy
        target.shape.assert_is_compatible_with(output.shape)

    ValueError: Shapes (None, 6) and (None, 8) are incompatible
```

In [41]:

```
print("Accuracy of our model on test data : " , model.evaluate(x_test,y_test)[1]*100 , "%")

epochs = [i for i in range(50)]
fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
test_acc = history.history['val_accuracy']
test_loss = history.history['val_loss']

fig.set_size_inches(20,6)
ax[0].plot(epochs , train_loss , label = 'Training Loss')
ax[0].plot(epochs , test_loss , label = 'Testing Loss')
ax[0].set_title('Training & Testing Loss')
ax[0].legend()
ax[0].set_xlabel("Epochs")
```

```
ax[1].plot(epochs , train_acc , label = 'Training Accuracy')
ax[1].plot(epochs , test_acc , label = 'Testing Accuracy')
ax[1].set_title('Training & Testing Accuracy')
ax[1].legend()
ax[1].set_xlabel("Epochs")
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [41], in <cell line: 1>()
----> 1 print("Accuracy of our model on test data : " , model.evaluate(x_test,y_test)[1]*100 , "%")
      3 epochs = [i for i in range(50)]
      4 fig , ax = plt.subplots(1,2)

File ~\anaconda3\lib\site-packages\keras\utils\traceback_utils.py:70, in filter_traceback.<locals>.error_handler(*args, **kwargs)
     67     filtered_tb = _process_traceback_frames(e.__traceback__)
     68     # To get the full stack trace, call:
     69     # `tf.debugging.disable_traceback_filtering()`
---> 70     raise e.with_traceback(filtered_tb) from None
     71 finally:
     72     del filtered_tb

File ~\AppData\Local\Temp\__autograph_generated_file4lk6hb0g.py:15, in outer_factory.<locals>.inner_factory.<locals>.tf__test_function(iterator)
     13 try:
     14     do_return = True
---> 15     retval_ = ag__.converted_call(ag__.ld(step_function), (ag__.ld(self), ag__.ld(iterator)), None, fscope)
     16 except:
     17     do_return = False

ValueError: in user code:

    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\training.py", line 1727, in test_function  *
        return step_function(self, iterator)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\training.py", line 1713, in step_function  **
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\training.py", line 1701, in run_step  **
        outputs = model.test_step(data)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\training.py", line 1667, in test_step
        self.compute_loss(x, y, y_pred, sample_weight)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\training.py", line 1052, in compute_loss
        return self.compiled_loss(
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\engine\compile_utils.py", line 265, in __call__
        loss_value = loss_obj(y_t, y_p, sample_weight=sw)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\losses.py", line 152, in __call__
        losses = call_fn(y_true, y_pred)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\losses.py", line 272, in call  **
        return ag_fn(y_true, y_pred, **self._fn_kwargs)
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\losses.py", line 1990, in categorical_crossentropy
        return backend.categorical_crossentropy(
    File "C:\Users\adity\anaconda3\lib\site-packages\keras\backend.py", line 5529, in categorical_crossentropy
        target.shape.assert_is_compatible_with(output.shape)

    ValueError: Shapes (None, 6) and (None, 8) are incompatible
```

In [42]:

```
# predicting on test data.
pred_test = model.predict(x_test)
y_pred = encoder.inverse_transform(pred_test)

y_test = encoder.inverse_transform(y_test)
```

```
36/36 [==============================] - 1s 20ms/step
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [42], in <cell line: 3>()
      1 # predicting on test data.
      2 pred_test = model.predict(x_test)
----> 3 y_pred = encoder.inverse_transform(pred_test)
      5 y_test = encoder.inverse_transform(y_test)

File ~\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:599, in OneHotEncoder.inverse_
transform(self, X)
    595 msg = (
    596     "Shape of the passed X data is not correct. Expected {0} columns, got {1}."
    597 )
    598 if X.shape[1] != n_transformed_features:
--> 599     raise ValueError(msg.format(n_transformed_features, X.shape[1]))
    601 # create resulting array of appropriate dtype
    602 dt = np.find_common_type([cat.dtype for cat in self.categories_], [])

ValueError: Shape of the passed X data is not correct. Expected 6 columns, got 8.
```

In [43]:
```
df = pd.DataFrame(columns=['Predicted Labels', 'Actual Labels'])
df['Predicted Labels'] = y_pred.flatten()
df['Actual Labels'] = y_test.flatten()

df.head(10)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [43], in <cell line: 2>()
      1 df = pd.DataFrame(columns=['Predicted Labels', 'Actual Labels'])
----> 2 df['Predicted Labels'] = y_pred.flatten()
      3 df['Actual Labels'] = y_test.flatten()
      5 df.head(10)

NameError: name 'y_pred' is not defined
```

In [44]:
```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize = (12, 10))
cm = pd.DataFrame(cm , index = [i for i in encoder.categories_] , columns = [i for i in encoder.cate
sns.heatmap(cm, linecolor='white', cmap='Blues', linewidth=1, annot=True, fmt='')
plt.title('Confusion Matrix', size=20)
plt.xlabel('Predicted Labels', size=14)
plt.ylabel('Actual Labels', size=14)
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [44], in <cell line: 1>()
----> 1 cm = confusion_matrix(y_test, y_pred)
      2 plt.figure(figsize = (12, 10))
      3 cm = pd.DataFrame(cm , index = [i for i in encoder.categories_] , columns = [i for i in enco
der.categories_])

NameError: name 'y_pred' is not defined
```

In [45]:
```
print(classification_report(y_test, y_pred))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [45], in <cell line: 1>()
----> 1 print(classification_report(y_test, y_pred))

NameError: name 'y_pred' is not defined
```

- We can see our model is more accurate in predicting surprise, angry emotions and it makes sense also because audio files of these emotions differ to other audio files in a lot of ways like pitch, speed etc..
- We overall achieved 61% accuracy on our test data and its decent but we can improve it more by applying more augmentation techniques and using other feature extraction methods.

This is all i wanna do in this project. Hope you guyz like this.

If you like the kernel make sure to upvote it please :-)