EDA-and-Data-Visualization

July 31, 2022

1 Introduction

The information in a speech is conveyed through words and emotions, meaning that in addition to the word, the way that it is said is important too. Therefore, in order for an artificial intelligence to understand the true meaning of a sentence, it needs to know the words and the way the words are said in the sentence. As a result, it is important to design a model to categorize the emotions in the audio files and speech.

In this work, by using the audio data provided by University of Toronto we are trying to design and train a model that can categorize different audio files according to the emotion that these files convey. This data set contains 2800 audio files categorized into 7 different "emotions" which are disgust, surprise, happy, sad, neutral, fear, and angry.

In this notebook, we will perform exploratory data analysis on these audio files by reading and converting them to numerical values by using liborsa, a python library designed to work with audio files. Then we will convert these audio files to mel-spectrogram to be used for our modeling.

2 Importing Libraries

[29]:
```
import pandas as pd
import os
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import librosa
import librosa.display
import noisereduce as nr
import IPython
from IPython.display import Audio

from sklearn.model_selection import train_test_split
from tqdm import tqdm

import warnings
warnings.filterwarnings('ignore')
```

3 Importing Data

In this section, we will import the data and we will save their name and the path that they are located in a dataframe.

[30]:
```
phase5_path = "/Users/miladshirani/Documents/Flatiron/phase_5"
base_path = "Emotional-Speech-Recognition"
toronto_data = "Toronto-Data"

data_path = os.path.join(phase5_path,base_path, toronto_data)
file_path = os.listdir(data_path)

files = {}
for i in file_path:
    file_i = os.path.join(data_path, i)
    files[i] = os.listdir(file_i)
```

[31]:
```
df = pd.DataFrame(list(zip(files.keys(), files.values())),
                  columns = ["data-name", "name"])

df = df.explode("name").reset_index(drop = True)

df["path"] = df.apply(lambda x: os.path.join(data_path,
                                              x["data-name"],
                                              x["name"]),
                                              axis = 1)

df["target"] = df["data-name"].apply(lambda x: x[4:].lower())
df.drop("data-name", inplace = True, axis = 1)

df["target"] = df["target"].apply(lambda x: "surprise"
                                  if x == "pleasant_surprise"
                                 or x == "pleasant_surprised"
                                 else x)
```

[32]:
```
df.target.value_counts(normalize = True)
```

[32]:
```
disgust     0.142857
surprise    0.142857
happy       0.142857
sad         0.142857
neutral     0.142857
fear        0.142857
angry       0.142857
Name: target, dtype: float64
```

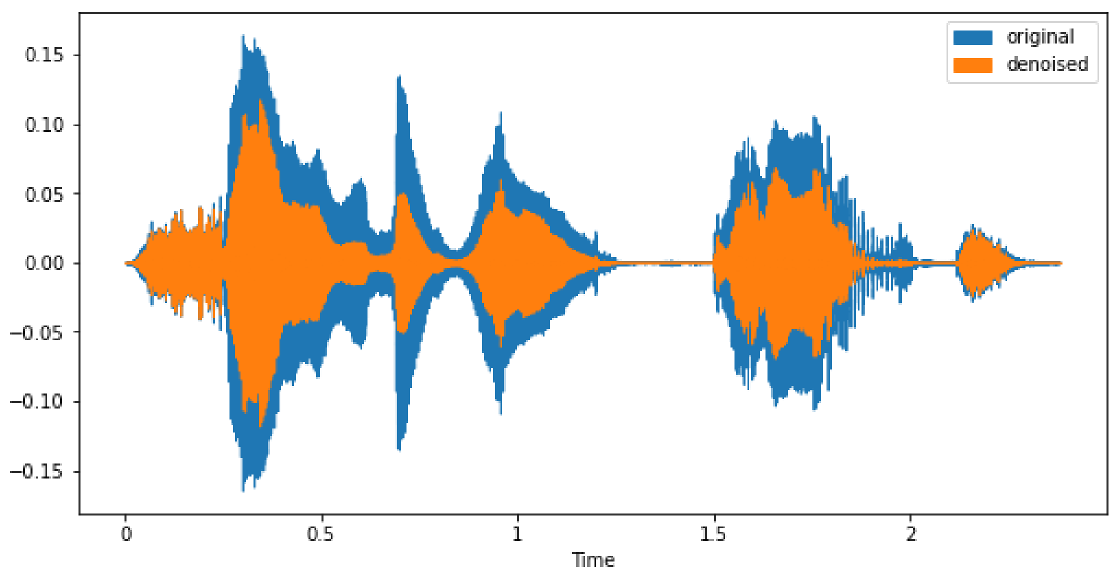
[33]:
```
len(df)
```

[33]: 2800

4 Some Basic Visualizations

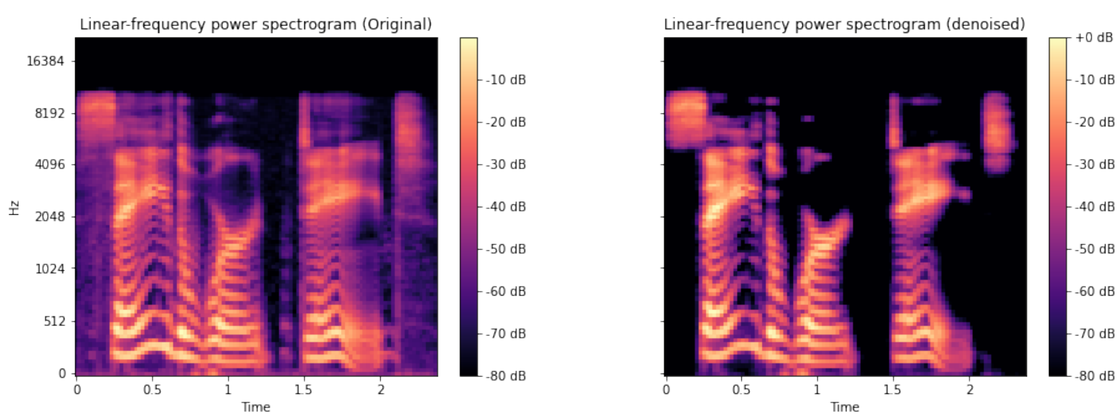
In this section, we will demonstrate an audio file (original and denoised versions) and its corresponding spectrogram.

[34]:
```
example_id = 0
audio_path = df["path"].iloc[example_id]
audio_target = df["target"].iloc[example_id]

x, sr = librosa.load(audio_path)
x_reduced = nr.reduce_noise(y=x, sr=sr)

plt.figure(figsize=(10, 5))
librosa.display.waveshow(x, sr=sr, label = "original")
librosa.display.waveshow(x_reduced, sr=sr, label = "denoised")
plt.legend();
```



[35]:
```
figs, axes = plt.subplots(ncols = 2, figsize = (15, 5))
figs.subplots_adjust(hspace=0.4, wspace=0.3)

hl = 512 # number of samples per time-step in spectrogram
hi = 100 # Height of image
wi = 384 # Width of image
fmax = sr
```

```
S = librosa.feature.melspectrogram(y=x, sr=sr,
                                   n_mels=hi, fmax=fmax,
                                   n_fft=2048, hop_length=hl)

S_reduced = librosa.feature.melspectrogram(y=x_reduced, sr=sr,
                                           n_mels=hi, fmax=fmax,
                                           n_fft=2048, hop_length=hl)

S_dB = librosa.power_to_db(S, ref=np.max)
S_reduced_dB = librosa.power_to_db(S_reduced, ref=np.max)



img_1 = librosa.display.specshow(S_dB, x_axis='time',
                         y_axis='mel', sr=sr,
                         fmax=fmax, ax = axes[0])
axes[0].set(title='Linear-frequency power spectrogram (Original)')
axes[0].label_outer()
figs.colorbar(img_1, ax=axes[0], format="%+2.f dB");

img_2 = librosa.display.specshow(S_reduced_dB, x_axis='time',
                         y_axis='mel', sr=sr,
                         fmax=fmax, ax = axes[1])
axes[1].set(title='Linear-frequency power spectrogram (denoised)')
axes[1].label_outer()
figs.colorbar(img_2, ax=axes[1], format="%+2.f dB");
```

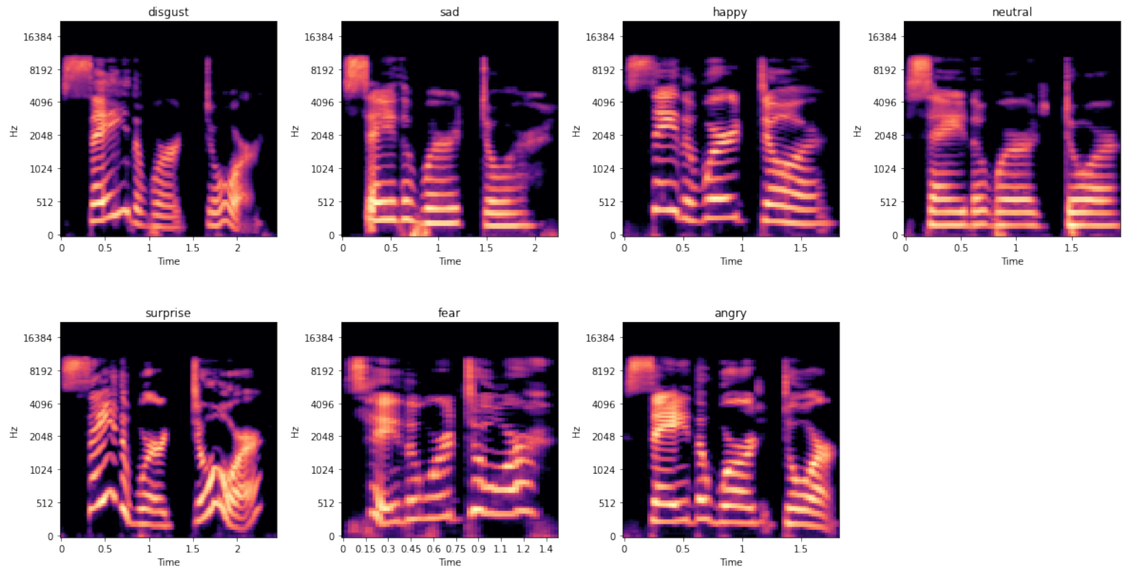


5 Effects of emotion in saying a word `door`

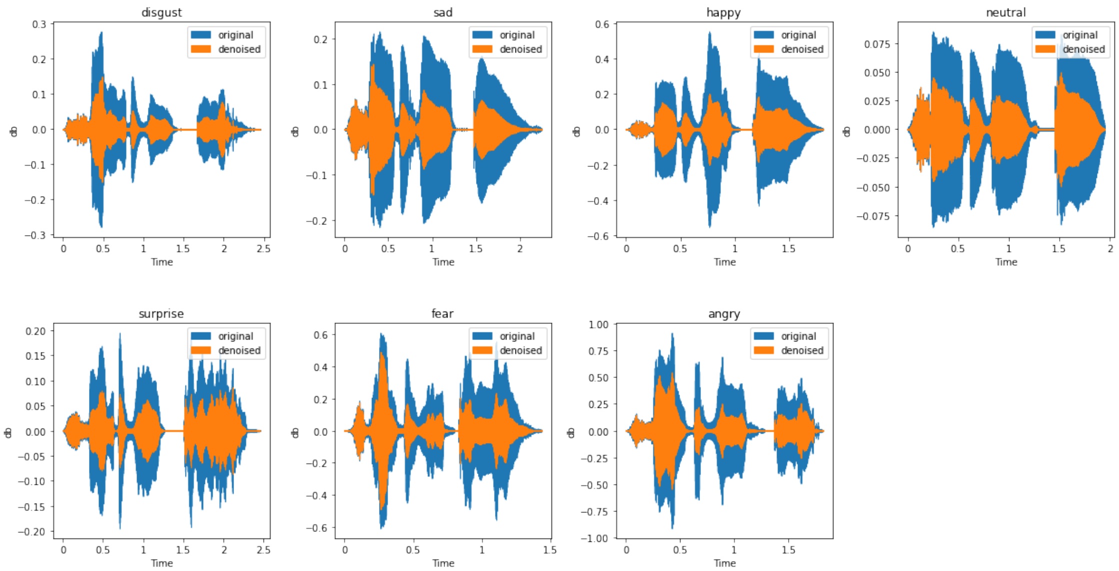
In this section, we will show different audio files and their spectrogram for different way of saying the word `door`, meaning that we want to see how saying a word affects the spectrogram of the audio file.

[36]:
```
door = df[["name", "target", "path"]].copy()
door["door"] = pd.DataFrame(door["name"].apply(lambda x: "YAF_door" in x).
 ↪astype(float))

emotion_door = door.loc[door["door"] == 1.0][["target", "path"]]
emotion_door.reset_index(inplace = True, drop = True)
```

[37]:
```
figs, axes = plt.subplots(nrows = 2 , ncols = 4, figsize = (20, 10))
figs.subplots_adjust(hspace=0.4, wspace=0.3)

for i in range(0, len(emotion_door)):

    ax = axes[i//4][i%4]
    audio_path = emotion_door.iloc[i]["path"]
    emotion = emotion_door.iloc[i]["target"]
    x, sr = librosa.load(audio_path)
    x_reduced = nr.reduce_noise(y=x, sr=sr)

    S_reduced = librosa.feature.melspectrogram(y=x_reduced,
                                               sr=sr,
                                               n_mels=hi,
                                               fmax=fmax,
                                               n_fft=2048,
                                               hop_length=hl)



    S_reduced_dB = librosa.power_to_db(S_reduced, ref=np.max)
    img_reduced = librosa.display.specshow(S_reduced_dB,
                                           x_axis='time',
                                           y_axis='mel',
                                           sr=sr,
                                           fmax=fmax,
                                           ax = ax)
    ax.set_title(emotion)

figs.delaxes(axes[1][3])
```



[38]:
```
figs, axes = plt.subplots(nrows = 2 , ncols = 4, figsize = (20, 10))
figs.subplots_adjust(hspace=0.4, wspace=0.3)

for i in range(0, len(emotion_door)):

    ax = axes[i//4][i%4]

    audio_path = emotion_door.iloc[i]["path"]
    emotion = emotion_door.iloc[i]["target"]
    x, sr = librosa.load(audio_path)
    x_reduced = nr.reduce_noise(y=x, sr=sr)
    librosa.display.waveshow(x, sr=sr, ax = ax, label = "original")
    librosa.display.waveshow(x_reduced, sr=sr, ax = ax, label = "denoised")
    ax.set_ylabel("db")
    ax.legend();
    ax.set_title(emotion)

figs.delaxes(axes[1][3])
```



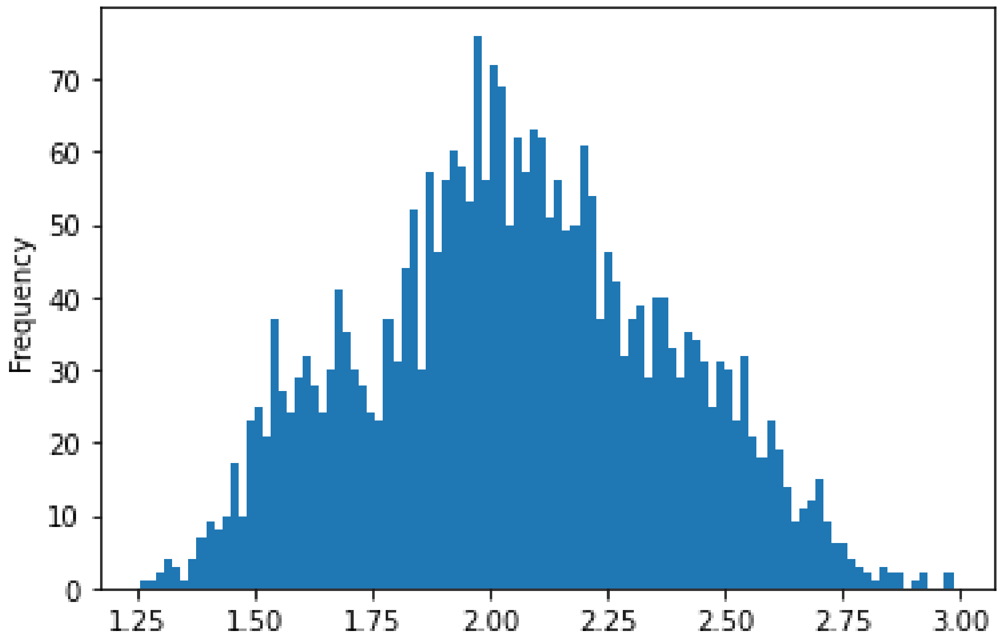
6 Length of audio files

In this section, we will plot the histogram of the length of audio files.

[42]:
```
def time(data):
    x, sr = librosa.load(data)
    return len(x)/sr

df["time"] = df["path"].apply(lambda x: time(x))
print("DONE")
```

```
DONE
```

[45]:
```
df["time"].plot(kind = "hist", bins = 100);
```



[46]:
```
df = df.sample(frac = 1)
df.to_csv("df.csv")
```

7 Train-Test-Split

In this section, we will split the data into train and test sets and we will save them for following notebooks.

[232]:
```
y = df["target"]
X = df[["path", "name"]]

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
                                                    test_size = 0.2,
                                                    random_state=42,
                                                    shuffle = True)


train = pd.concat([X_train, y_train], axis = 1)
test = pd.concat([X_test, y_test], axis = 1)

train.to_csv("../Train-Test-Split/train.csv")
test.to_csv("../Train-Test-Split/test.csv")
```

8 Mel Spectrogram

In this section, we will create and save the mel-spectrogram of the train and test sets and will save them for the following notebooks.

[267]:
```
def mel_spectrogram(target, data, path, sr = 22050):

    hl = 512
    hi = 100
    wi = 384
    fmax = sr

    phase5_path = "/Users/miladshirani/Documents/Flatiron/phase_5/Emotional␣
↪Speech Recognition"
    mel_spectrogram = "mel_spectrogram"

    target_path = os.path.join(phase5_path,
                               mel_spectrogram,
                               path, target)

    data_set = data[data["target"] == target]

    for i in tqdm(range(len(data_set))):

        audio_path = data_set["path"].iloc[i]
        audio_name = data_set["name"].iloc[i][:-4]

        x, _ = librosa.load(audio_path)
        x_reduced = nr.reduce_noise(y=x, sr=sr)

        S = librosa.feature.melspectrogram(y=x_reduced, sr=sr, n_mels=hi,
                                           fmax=fmax, hop_length=hl)

        S_dB = librosa.power_to_db(S, ref=np.max)

        img = librosa.display.specshow(S_dB,
                                       x_axis='mel',
                                       y_axis='time',
                                       sr=sr,
                                       fmax=fmax)


        mel_name = audio_name + ".png"
        plt.tick_params(left = False, right = False , labelleft = False ,
                labelbottom = False, bottom = False)
        a = plt.gca()
        # set visibility of x-axis as False
        xax = a.axes.get_xaxis()
```

```python
        xax = xax.set_visible(False)

        # set visibility of y-axis as False
        yax = a.axes.get_yaxis()
        yax = yax.set_visible(False)

        plt.ioff()
        plt.savefig(fname = os.path.join(target_path, mel_name),
                    dpi = 400,
                    bbox_inches = "tight",
                    pad_inches = 0)


        plt.close()


    print("DONE!")
```

[ ]:
```python
targets = list(train["target"].unique())
```

[269]:
```python
for item in targets:
    print(f"{item}\n")
    mel_spectrogram(target = item, data = train, path = "train", sr = 22050)
```

```
  0%|          | 0/320 [00:00<?, ?it/s]

angry


100%|     | 320/320 [01:50<00:00,  2.89it/s]
  0%|          | 0/320 [00:00<?, ?it/s]

DONE!
surprise


100%|     | 320/320 [01:49<00:00,  2.91it/s]
  0%|          | 0/320 [00:00<?, ?it/s]

DONE!
neutral


100%|     | 320/320 [01:51<00:00,  2.86it/s]
  0%|          | 0/320 [00:00<?, ?it/s]

DONE!
fear
```

```
100%|      | 320/320 [01:44<00:00,  3.06it/s]
  0%|          | 0/320 [00:00<?, ?it/s]

DONE!
disgust

100%|      | 320/320 [01:54<00:00,  2.81it/s]
  0%|          | 0/320 [00:00<?, ?it/s]

DONE!
sad

100%|      | 320/320 [01:54<00:00,  2.79it/s]
  0%|          | 0/320 [00:00<?, ?it/s]

DONE!
happy

100%|      | 320/320 [01:52<00:00,  2.83it/s]

DONE!
```

[270]:

```
for item in targets:
    print(f"{item}\n")
    mel_spectrogram(target = item, data = test, path = "test", sr = 22050)
```

```
  0%|          | 0/80 [00:00<?, ?it/s]

angry

100%|      | 80/80 [00:26<00:00,  3.02it/s]
  0%|          | 0/80 [00:00<?, ?it/s]

DONE!
surprise

100%|      | 80/80 [00:27<00:00,  2.95it/s]
  0%|          | 0/80 [00:00<?, ?it/s]

DONE!
neutral

100%|      | 80/80 [00:26<00:00,  2.98it/s]
  0%|          | 0/80 [00:00<?, ?it/s]
```

```
DONE!
fear

100%|          | 80/80 [00:25<00:00,  3.11it/s]
  0%|          | 0/80 [00:00<?, ?it/s]

DONE!
disgust

100%|          | 80/80 [00:29<00:00,  2.67it/s]
  0%|          | 0/80 [00:00<?, ?it/s]

DONE!
sad

100%|          | 80/80 [00:29<00:00,  2.71it/s]
  0%|          | 0/80 [00:00<?, ?it/s]

DONE!
happy

100%|          | 80/80 [00:29<00:00,  2.73it/s]

DONE!
```

[272]:
```
targets = list(train["target"].unique())
targets
```

[272]:
```
['angry', 'surprise', 'neutral', 'fear', 'disgust', 'sad', 'happy']
```

9 Next

In the following notebooks, we will train several ML and NN models on the numerical values of the audio files and the spectrograms.