

IAT481 Assignment 2: Speech Emotion Recognition Dataset Analysis

Wei Xing Deng #301442155

Word count: 746

School of Interactive Arts and Technology, Simon Fraser University

IAT 481W D100: Exploring Artificial Intelligence: Its Use, Concepts, and Impact

Professor: Dr. O. Nilay Yalcin

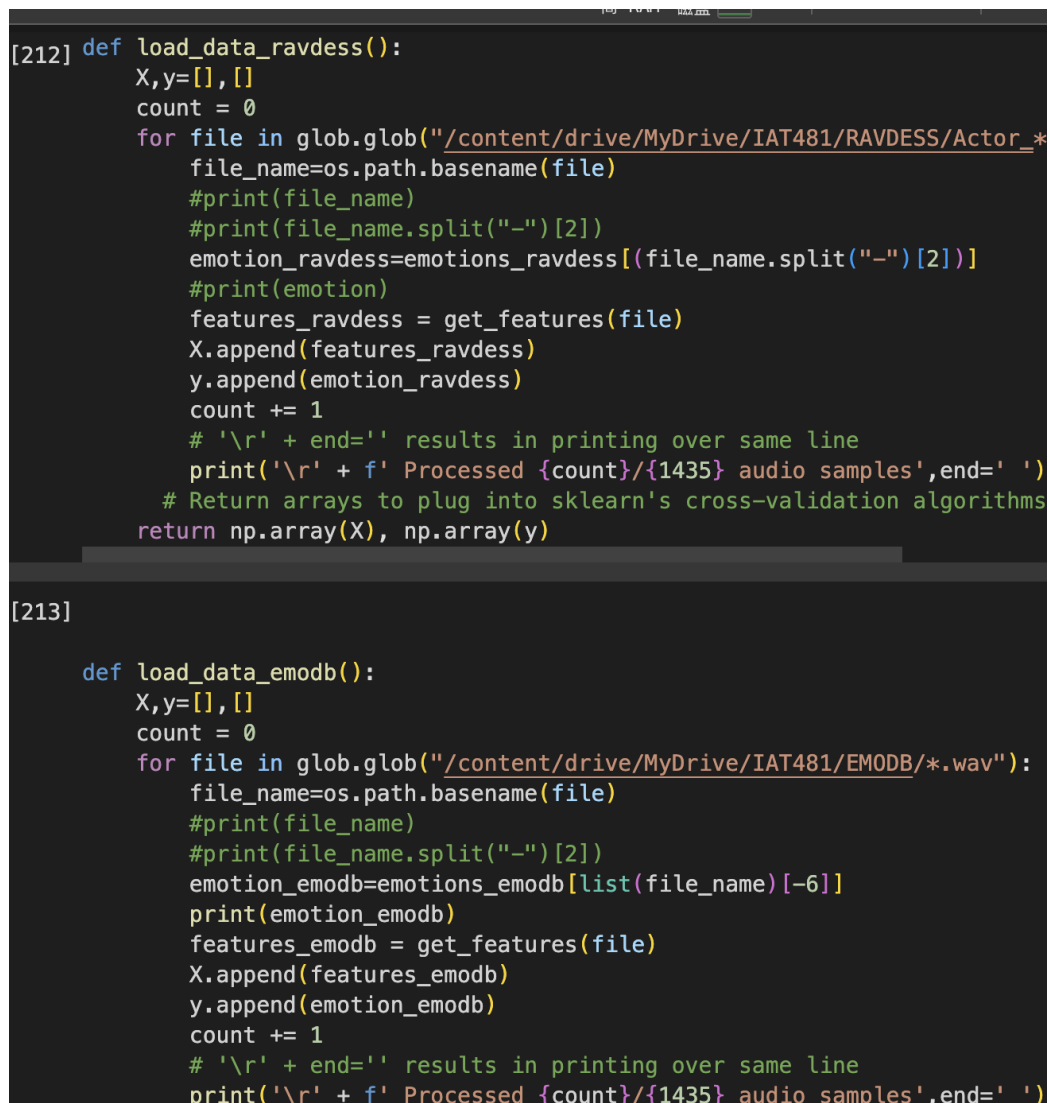
TA: Maryiam Zahoor

Jan 25, 2024,

I was confused about what to do and how to use Colab at the beginning of this assignment. Luckily, Maryiam helped me during office hours. Thus, I figured out most of the parts and what I should do for this assignment. However, for some of the problems, like gender balance and emotional category balance, I am not sure whether both balances should come from one dataset or not, which I will mention later. Another problem would be the data cleaning process. Should I solely use feature scaling, or should I look up on the internet and find my solution to clean the data? Other than These, I have tackled the tasks.

The solution I approached was to use the emotion dictionaries for both RAVDESS and EmoDB. I loaded the emotional data from each dataset separately, because of the different naming conventions they had after converting them into the data frames and displaying them in the bar chart.

Figure.1



```
[212] def load_data_ravdess():
    X,y=[],[]
    count = 0
    for file in glob.glob("/content/drive/MyDrive/IAT481/RAVDESS/Actor_*"):
        file_name=os.path.basename(file)
        #print(file_name)
        #print(file_name.split("-")[2])
        emotion_ravdess=emotions_ravdess[(file_name.split("-")[2])]
        #print(emotion)
        features_ravdess = get_features(file)
        X.append(features_ravdess)
        y.append(emotion_ravdess)
        count += 1
        # '\r' + end='' results in printing over same line
        print('\r' + f' Processed {count}/{1435} audio samples',end=' ')
    # Return arrays to plug into sklearn's cross-validation algorithms
    return np.array(X), np.array(y)

[213]

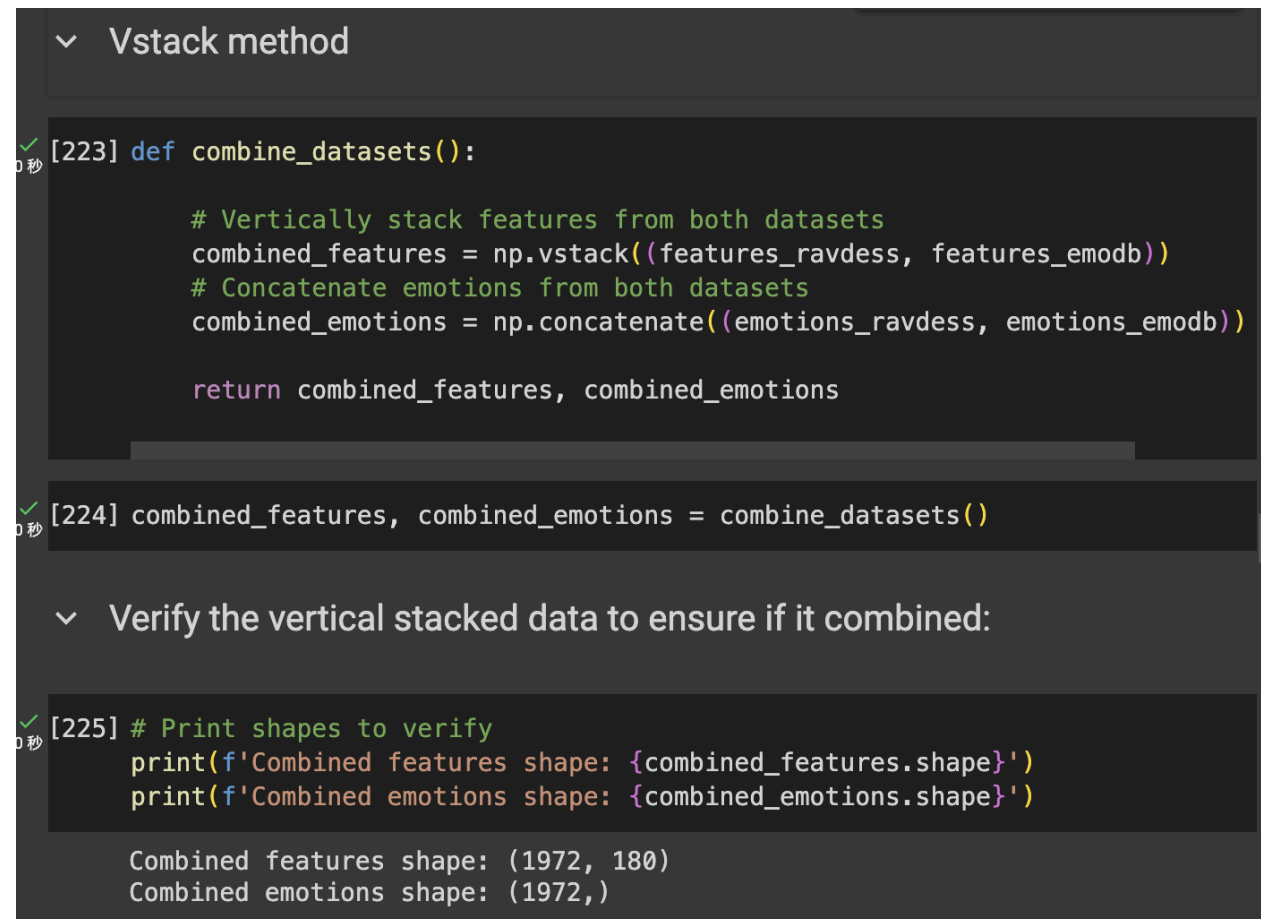
def load_data_emodb():
    X,y=[],[]
    count = 0
    for file in glob.glob("/content/drive/MyDrive/IAT481/EMODB/*.wav"):
        file_name=os.path.basename(file)
        #print(file_name)
        #print(file_name.split("-")[2])
        emotion_emodb=emotions_emodb[list(file_name)[-6]]
        print(emotion_emodb)
        features_emodb = get_features(file)
        X.append(features_emodb)
        y.append(emotion_emodb)
        count += 1
        # '\r' + end='' results in printing over same line
        print('\r' + f' Processed {count}/{1435} audio samples',end=' ')
```

Note. From Screenshot of Colab [Screenshot], by D. Weixing, 2024,

Finally, I combined them with the Vstack and Concatenate methods. I vertically stacked both features_ravdess and features_emodb and concatenated both emotions_ravdess and emotions_emodb (Figure 2). After combining the datasets, I found mismatching categories such as 'calm', 'boredom', and 'surprised'. In this case, I took the approach from what Maryiam taught in the lab, removing samples that both datasets have not in common (Figure 3).

Figure.2

Vstack and Concatenate:



```

▼ Vstack method

[223] def combine_datasets():

    # Vertically stack features from both datasets
    combined_features = np.vstack((features_ravdess, features_emodb))
    # Concatenate emotions from both datasets
    combined_emotions = np.concatenate((emotions_ravdess, emotions_emodb))

    return combined_features, combined_emotions

[224] combined_features, combined_emotions = combine_datasets()

▼ Verify the vertical stacked data to ensure if it combined:

[225] # Print shapes to verify
print(f'Combined features shape: {combined_features.shape}')
print(f'Combined emotions shape: {combined_emotions.shape}')

Combined features shape: (1972, 180)
Combined emotions shape: (1972,)
```

Note. From Screenshot of Colab [Screenshot], by D. Weixing, 2024,

Figure. 3

Handling mismatches categories:

```
[228] # combined_features, combined_emotions are as obtained from the combine_datasets function

# Find indices of samples that are not 'calm' or 'boredom'
indices_to_keep = np.where((combined_emotions != 'calm') & (combined_emotions != 'boredom'
& (combined_emotions != 'surprised')))[0]

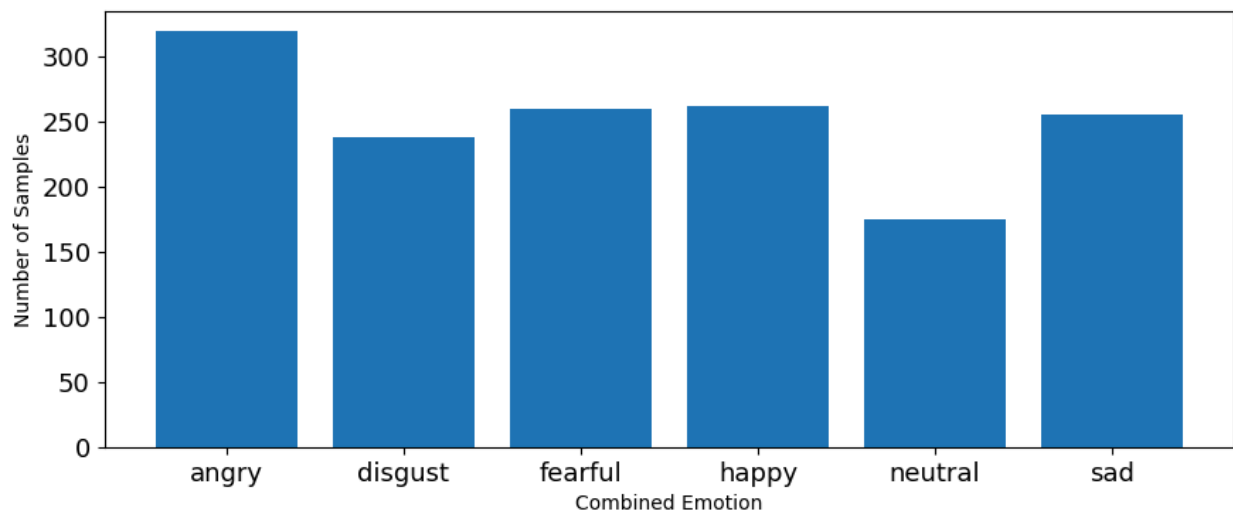
# Filter both features and emotions arrays
filtered_features = combined_features[indices_to_keep]
filtered_emotions = combined_emotions[indices_to_keep]

print(f'Filtered features shape: {filtered_features.shape}')
print(f'Filtered emotions shape: {filtered_emotions.shape}')

Filtered features shape: (1509, 180)
Filtered emotions shape: (1509,)
```

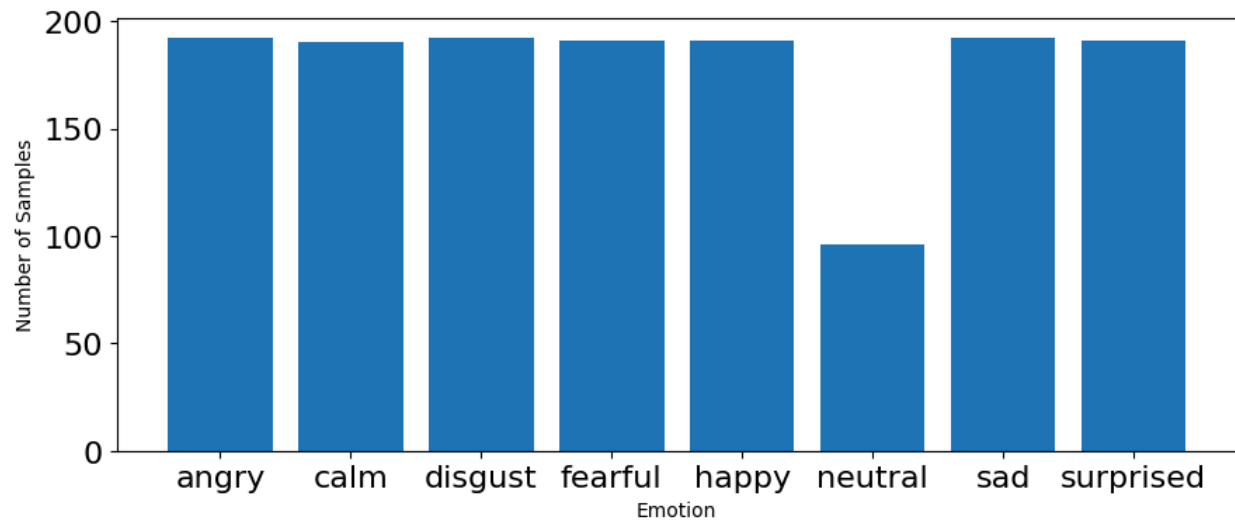
Note. From Screenshot of Colab [Screenshot], by D. Weixing, 2024,

Here is what I have displaying the Combined Emotion Bar Chart:



In the combined emotion dataset, Angry samples are slightly above other samples. Neutral, on the other hand, is below the average samples. A possible impact would be the accuracy of the model using this dataset because each sample is not even out. Especially the 'angry' and 'neutral' ones.

Here is the Ravdess-only Emotion Bar Chart:



In comparison with the combined dataset, Ravdess emotion's data, excluding the neutral sample, contains the same number of samples that are great for model training. On the other hand, EmoDB contains a greater number of angry samples, which makes the combined dataset unbalanced.

Data Cleaning (ChatGPT):

I didn't quite understand how to clean the data properly. Throughout the lecture, we were told to clean missing or duplicate data from the dataset. Therefore, I asked chatGPT, and it provided me with various solutions on how to clean the dataset effectively via a small chunk of codes (Figure 4).

Figure. 4

Removing missing and duplicate values.

```
[247]
# Check for NaN values in the array
has_nan = np.isnan(filtered_features).any()

print(f"Does the combined features array contain NaN values? {has_nan}")

Does the combined features array contain NaN values? False

[248]
# Find unique rows and indices to identify duplicates
_, unique_indices = np.unique(filtered_features, axis=0, return_index=True)

# Calculate the number of duplicates
num_duplicates = filtered_features.shape[0] - len(unique_indices)

print(f"Number of duplicate rows: {num_duplicates}")

# Optional: To remove duplicates
unique_features = filtered_features[unique_indices]
print(f"Array shape after removing duplicates: {unique_features.shape}")

Number of duplicate rows: 2
Array shape after removing duplicates: (1507, 180)
```

Note. From Screenshot of Colab [Screenshot], by D. Weixing, 2024,

Feature scaling:

The rest would be similar to the tutorial code.

Standard Scaling:

12 Chromagram features: min = -4.048, max = 2.602, mean = 0.000, deviation = 1.000

128 Mel Spectrogram features: min = -0.475, max = 33.103, mean = 0.000, deviation = 1.000

40 MFCC features: min = -4.720, max = 6.584, mean = 0.000, deviation = 1.000

MinMax Scaling:

12 Chromagram features: min = 0.000, max = 1.000, mean = 0.599, deviation = 0.179

128 Mel Spectrogram features: min = 0.000, max = 1.000, mean = 0.024, deviation = 0.068

40 MFCC features: min = 0.000, max = 1.000, mean = 0.393, deviation = 0.177

Gender distributions and balances

I think I fried my brain too much and did not think about having a three-dimensional array to store the gender data within the combined emotion dataset. Instead, I create another method for both RAVDESS and EmoDB to filter out genders from each sample, which means I create another two dimension dataset other than the combined emotion dataset.

Figure. 5

Gender filter method.

```
[235] def load_data_ravdess_gender():
    X,y=[],[]
    count = 0
    for file in glob.glob("/content/drive/MyDrive/IAT481/RAVDESS/Actor_*/*.wav"):
        file_name = os.path.basename(file)
        # Attempt to extract the actor number more robustly
        actor_number_str = file_name.split("-")[-1].split(".")[0]
        # Filter out non-digit characters to ensure a valid integer conversion
        actor_number_str = ''.join(filter(str.isdigit, actor_number_str))
        if actor_number_str: # Ensure string is not empty
            actor_number = int(actor_number_str)
            # Determine gender based on the actor number
            gender = 'male' if actor_number % 2 != 0 else 'female'
            features = get_features(file)
            X.append(features)
            y.append(gender)
            count += 1
            print('\r' + f'Processed {count}/{1435} audio samples', end=' ')
        else:
            print(f"\nWarning: Could not determine actor number from file name '{file_name}'")
    return np.array(X), np.array(y)
```

Note. From Screenshot of Colab [Screenshot], by D. Weixing, 2024,
Instead of using the dictionary, I choose a better way to identify gender via modulus %. If there is no reminder, it's male. If there is a reminder, it's female.

One of the problems that I encountered was when the file was processed to 313/1435, it appeared to be an error saying the invalid literal for int() with base 10: '10 (1)'.

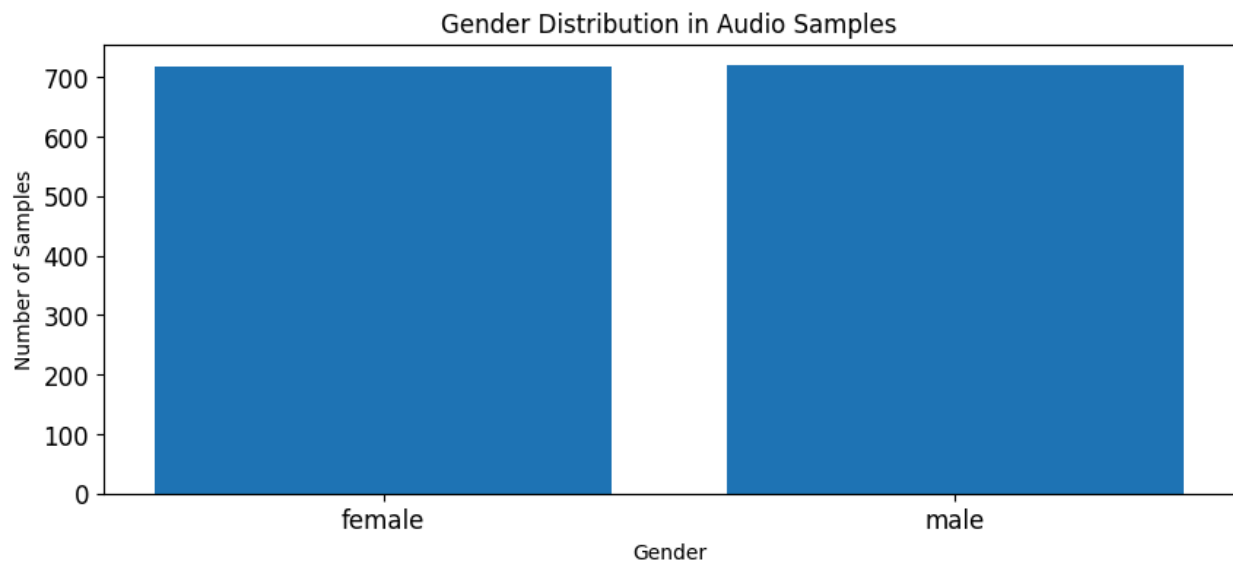
I assumed it was a duplicate file. Thus, I did a file search.

I checked all the files and cannot locate any of them that have 10(1) at the end of their file name. Therefore, I ask chatGPT to solve my problem.

[Source] chatGPT prompt:

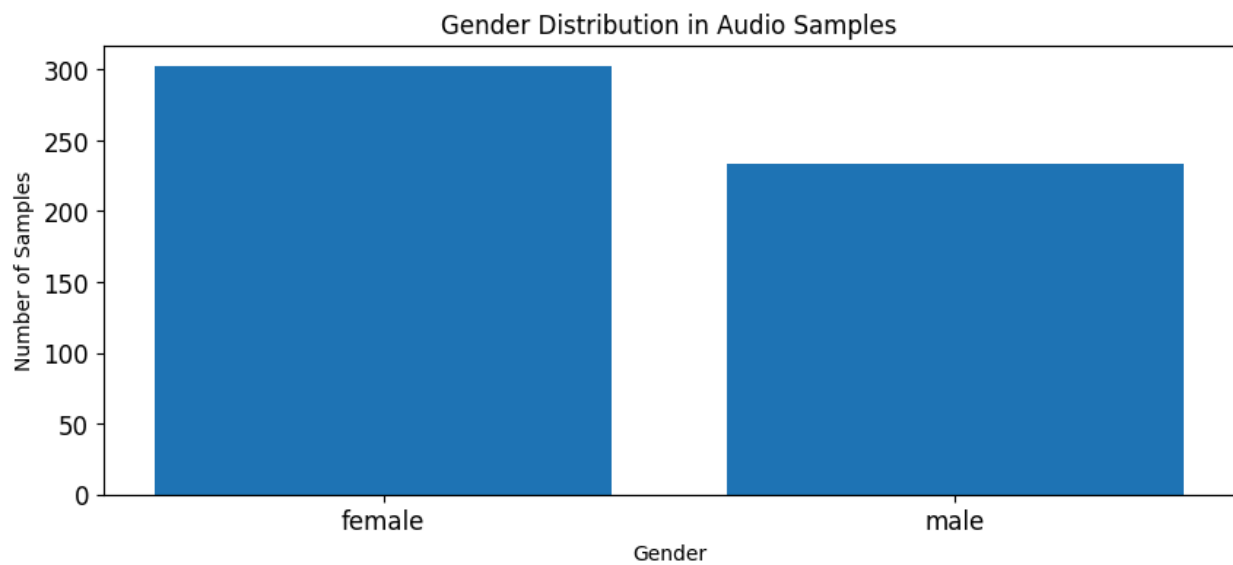
- `features_ravdess, gender_ravdess = load_data_ravdess_gender()`
- Output: `ValueError: invalid literal for int() with base 10: '10 (1)'`

Here is the Gender Distribution of RAVDESS:



The Gender distribution of RAVDESS is balanced, both shares around 700 samples in the dataset.

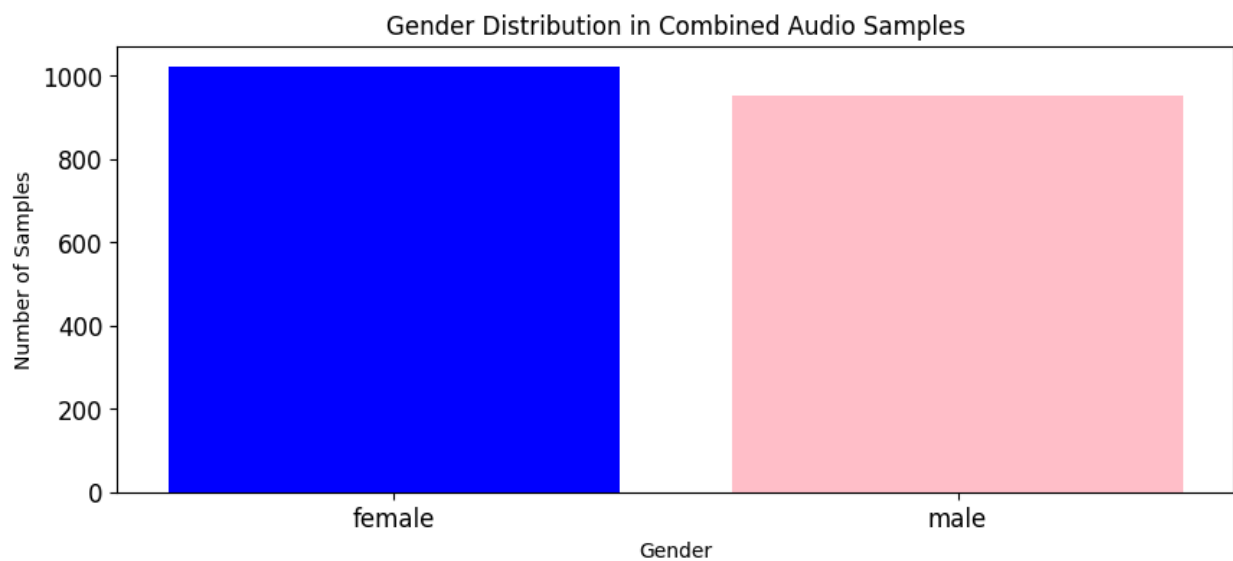
Here is the Gender Distribution of EmoDB:



The male is significantly less than the female samples in the EmoDB dataset.

Here is the Combined dataset of both RAVDESS and EmoDB:

After combining both features vs genders datasets, here is what I have got for the gender distribution.



There are about 70 fewer actors compared with females. Whereas in the RAVDESS dataset, the male and female distributions are even. In the combined dataset, this might lead to the consequence of inaccuracy when detecting male voices compared to females.

Reference

Python Lists. (2024). W3schools.com. https://www.w3schools.com/python/python_lists.asp

Python 2.7 Tutorial. (2024). Pitt.edu. <https://sites.pitt.edu/~naraehan/python2/tutorial9.html>

OpenAI. (2023). *ChatGPT* (Mar 14 version) [Large language model].
<https://chat.openai.com/chat>