Homework 3: Emotion Recognition

For this assignment, you will work on emotion recognition in speech. The speech segments are extracted from the Emotional Prosody Speech and Transcripts (https://catalog.ldc.upenn.edu/LDC2002S28). In the speech files folder, there are 2,324 WAV files, and all files are named with the format "speaker_session_emotion_start-time_content.wav". These files are from 7 speakers (cc, cl, gg, jg, mf, mk, mm), labeled with 15 emotions (anxiety, boredom, cold-anger, contempt, despair, disgust, elation, happy, hot-anger, interest, neutral, panic, pride, sadness, shame).

What to submit

Submit a zipped folder <uni.zip> containing the following files to Gradescope.

1. Feature extraction and analysis script (.ipynb)
2. Classification script (.ipynb)
3. Report with your responses to the questions in the following 3 sections (.pdf (preferred), .doc, .docx)
    a. Feature analysis
    b. Classification experiments
    c. Error analysis
4. README/documentation of your code (.README (preferred), .txt)

Notes

1. Please retain your outputs in the IPython notebooks.
2. You don't have to submit any interim/output data files or feature files, but please make sure that all features used in the experiments can be reproduced by running your code.
3. This assignment will be graded based on your code and report. We will check the code and ensure that it is consistent with your report. Make sure that your report is easily reproducible from your code -- include any necessary instructions to run your code in your README.
4. **Please add this statement to your report (*.pdf (preferred), *.doc, *.docx): "I confirm that I have not used any GPT-generated response for any part of this homework."**

Task 0: GPT-use affirmation

Please confirm your compliance by typing the following statement at the top of your report: 'I confirm that I have not used any GPT-generated responses for any part of this assignment.'

1. Feature Analysis (40 points)

Extract six features from each speech segment:

- The min, max, mean of pitch
- The min, max, mean of intensity

Resources:

Praat
Parselmouth (a Python library for Praat)

Note: For Praat-based pitch extraction, please set the pitch range as 75~600 Hz, and use autocorrelation as the analysis method. For intensity extraction, please set the pitch floor to 75 Hz, and the time step to 0.0 or None. Please use only the left channel (channel 1) for analysis in this section.

Since each speaker naturally has a different pitch range and other voice qualities like intensity, you need to normalize the features accordingly by speaker. There are at least two ways you may want to normalize:

1. Z-score normalization (standardization) over the individual speaker is one common method. For example, if you would like to normalize the pitch features of an individual speaker X, the steps to Z-score normalization can be:
    a. Extract **raw** pitch values for all speech segments from speaker X. This would result in one array of pitch/frequency values for each segment. You can get it by calling
    `pitch_array = pitch.selected_array["frequency"]`
    b. Concatenate all pitch arrays from speaker X and calculate an **overall** mean pitch (μ_X) and pitch std (σ_X) of speaker X.
    c. Normalize each extracted pitch array – for each value $\boldsymbol{x}$ in an array, calculate the normalized $\boldsymbol{x}$ as ($\boldsymbol{x}$ - μ_X) / σ_X.
    d. Finally, calculate the min, max, and mean of pitch for each segment using the normalized pitch array.
    Note: You are supposed to obtain an overall pitch/intensity mean and std value for each **speaker instead of each **feature**, in order to normalize by the speaker. Do **not** get a mean value of mean pitches, rather, get an **overall** mean value of **all pitch values** from all segments*

2. Another method you may want to try is normalizing by the means of the individual speaker's neutral utterances. For example, to normalize the min pitch values of an individual speaker X,
    a. Extract pitch values for all speech segments from speaker X, and obtain a min pitch value for each.
    b. Calculate an overall mean of min pitch ($\mu_{minpitch\_X}$) of speaker X by averaging the min pitch values of all neutral segments from speaker X.
    c. Normalize each min pitch value – for each min pitch $x_{\boldsymbol{min}}$ of a segment, calculate the normalized $x_{\boldsymbol{min}}$ as $x_{\boldsymbol{min}}$ - $\mu_{minpitch\_X}$.

You can either pick one of the normalization methods above or use other methods you find helpful. Please specify your method and provide a detailed description of how you calculated it and why you chose it. ***You should exclude values of pitch and intensity that are 0 or NaN.***

You need to turn in plots of the mean and standard deviation of each feature for all of the 15 emotion classes. Please also specify for each plot whether it was created a) without normalization; b) with normalization (tell us what normalization method you used, how you calculated it, and why you chose this method). Specifically, create 2 plots for each feature, one without normalization, and one with normalization. In each plot, visualize the emotion-level mean and standard deviation values (calculated using all speech files of a certain emotion) for all emotion classes. You can use graphs with error bars to visualize the mean and standard deviation, as illustrated in the examples in Figure 1, where the x-axes can be emotion classes, and the y-axes can be any one of the 6 feature values (e.g. min pitch, mean intensity, etc.). This will result in 12 (6 x 2) plots.

In addition, tell us what you learn from these plots. **Please report and discuss at least 5 interesting observations.**

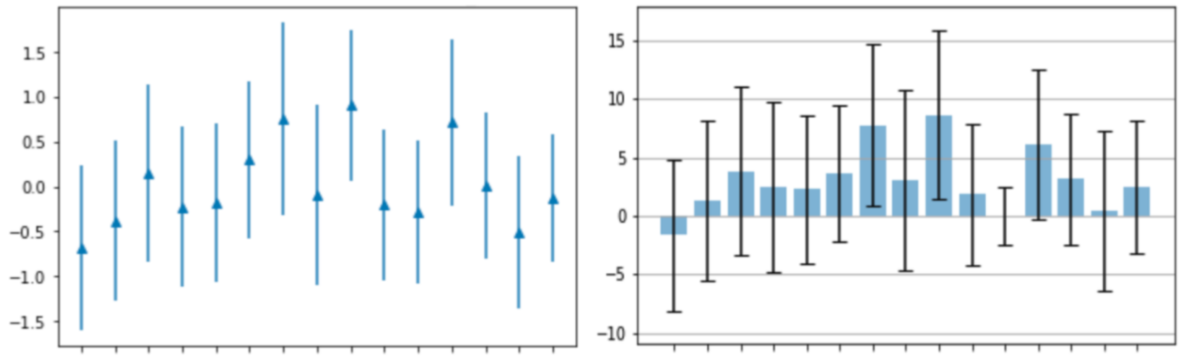

Figure 1: Examples of error bars

2. Classification Experiments (30 points)

Extract a set of acoustic-prosodic features using the openSMILE toolkit. Normalize your extracted features (as in Part 1.Feature Analysis) and use leave-one-speaker-out cross-validation to predict the emotion classes. Leave-one-speaker-out cross-validation means, for each speaker S, train on all other six other speakers combined and test on S.

Report the classification results (screenshot or copy-paste sklearn classification reports) for all 7 experiments (leave one speaker out as the test set in each experiment). Also, compute and report aggregated average accuracy and **weighted** F1 scores over all the experiments and emotions. **The aggregated metrics should be calculated as follows:**

$$\boldsymbol{aggregated\ average\ accuracy} = \sum_{i=1}^{7} (\boldsymbol{acc_i} \cdot \boldsymbol{n_i}) \,/\, \sum_{i=1}^{7} \boldsymbol{n_i}$$

$$\boldsymbol{aggregated\ average\ F1} = \sum_{i=1}^{7} (\boldsymbol{F1_i} \cdot \boldsymbol{n_i}) \,/\, \sum_{i=1}^{7} \boldsymbol{n_i}$$

Here, acc_i and $F1_i$ represent the accuracy and **weighted** F1 scores for the test set of speaker S in each run, and n_i is the number of samples in the corresponding test set.

Resources:
openSMILE
Scikit-learn (sklearn)
Note: Please check the documentation page for detailed installation instructions:
Here is a quick guide:

```
git clone https://github.com/audeering/opensmile.git
cd opensmile/
bash build.sh
```

Feature extraction: under `opensmile` directory, run

```
./build/progsrc/smilextract/SMILExtract -C config_path -I input_path -O
output_path
```

We recommend using the configuration file provided in the toolkit distribution to extract “The INTERSPEECH 2009 Emotion Challenge feature set” as a start. The config file can be found at

```
./config/is09-13/IS09_emotion.conf
```

You may supplement the openSMILE features with **additional** features, such as textual features or other speech-related features. However, please note that the **openSMILE features must be included**. Ensure that you provide a clear justification for your feature selection and include a detailed description of all features in your report.

You can train a multiclass classifier and use the classification report function in sklearn to generate the results. Regarding the classifier, you can use either a traditional machine learning model, such as random forest and SVM, or a neural network model. Report the type and structure of the model you use. Please avoid excessive tuning of the hyperparameters of the classifier you use, since you want to avoid overfitting the dataset.

Notes: try your best for the model performance and report the best aggregated average accuracy and aggregated average **weighted** F1 score.

3. Error analysis (30 points)

Analyze the errors made by your best performing leave-one-speaker-out experiment, i.e. the best results you got for one of the 7 speakers.

What do you observe from the results you got for this speaker overall? And, in more detailed observations, which class(es) were easiest to predict? Why do you think they were easy? Which were the most difficult? Why do you think they were difficult? Based on this analysis, what ideas do you have to further improve your classifier?