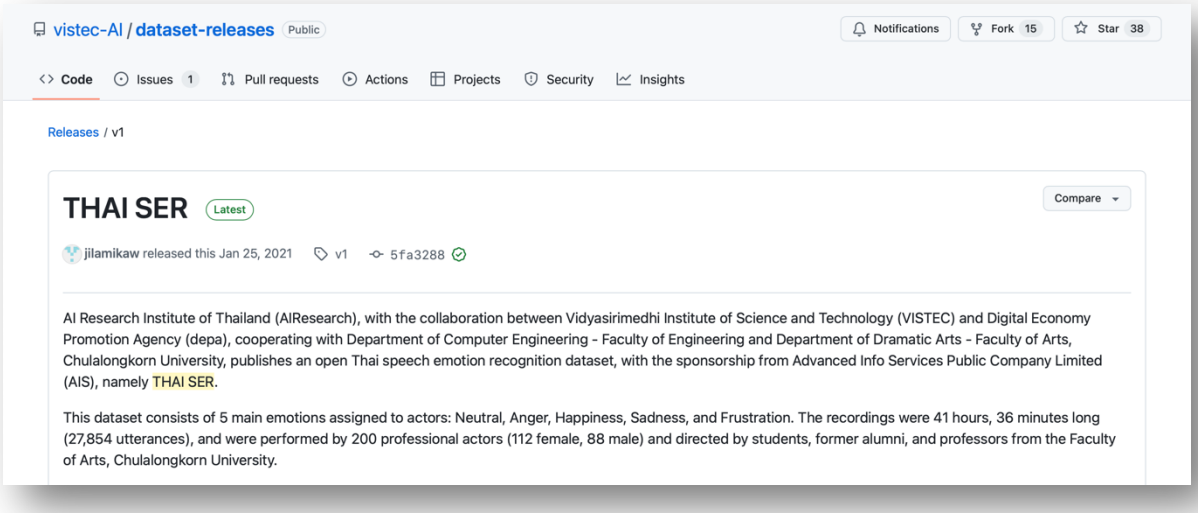
Cloudy's OS:
AUDIO EMOTION ANALYSIS

ITCS225 Principles of Operating System – Group Project

- 6688077 Bhumipat Pengpaiboon
- 6688108 Napas Siripaskrittakul
- 6688142 Krerkkiat Wattanaporn

1. Introduction
   1.1. This report is focused on the implementation and performance analysis of a system designed to classify emotions from spoken Thai language. The main objective key is developing a machine learning to learn and identifying five emotional by the sound via dataset with 5 different classes such as angry, frustrated, happy, neutral, and sad from THAI SER dataset. We are trying to examine the performance and monitor the result by looking on involving data loading, feature extraction, model training, and evaluation as a measured time. This report will evaluate the predict accuracy and the performance between first-run version and enhancement version.

2. Methodology
   2.1. Dataset: The THAI SER dataset was developed by AI Research Thailand in collaboration with Chulalongkorn University and AIS. In the dataset, there are 41.6 hours (25,185 audio files used in this analysis) of Thai speech audio recordings from 200 actors, annotated with the five target emotions.



2.2. Feature Extraction: We are using Mel-Frequency Cepstral Coefficients (MFCCs) to analyze the audio for the computer to understand by transferring from sound as a frequency in Hz to pure data by capturing information about the shape of the vocal. Each audio file sampled at 44.1 kHz with 13 MFCCs were extracted using the **librosa library**. To transform the input data to machine learning part, we are needed to checking input dimensionality for the machine learning model so the resulting feature sequences were standardized to a fixed length of 500-time steps by padding shorter sequences. The 2D feature array (500 steps x 13 MFCCs) was flattened into a 1D vector of 6500 features per audio sample.

2.3. Model: A Multi-Layer Perceptron (MLP) classifier is implemented by using **scikit-learn (MLPClassifier).** We are also splitting the data for training (70%), validation (15%), and test (15%) sets, followed by feature scaling using by **StandardScaler**.

2.4. The versions: We have two versions of code on this implementation to compare the computational efficiency on performance and result.

- The first version: This version reading the dataset sequentially and doing the Feature extraction that keep iterating through each audio file path in the metadata and transform it into the DataFrame from NumPy library and calling the **extract_features function** for each file within a standard for loop. This make the code to utilizes a single CPU core for the feature extraction process which is slow.

```
X_list = []
# ...
for index, row in tqdm(df_meta.iterrows()):
    features = extract_features(row['path'])
    X_list.append(features.flatten())
X = np.array(X_list)
```

- The final version: This version is using the parallel computation which is enable the code to run by the parallel processing with hardware accelerate from **joblib library** and **joblib.Parallel utility** so these function is calling the **extract_features** function to using multiple available CPU cores as (n_jobs=-1) to processing multiple audio files **concurrently**.

```
filepaths = df_meta['path'].tolist()
results = Parallel(n_jobs=-1, backend='loky')(
    delayed(extract_features)(fp) for fp in tqdm(filepaths)
)
X_list = [res.flatten() for res in results]
X = np.vstack(X_list)
```

For metadata processing, preprocessing, training, and evaluation is remained sequential in both implementations because it does not affect to significant of the performance and result.

2.5. Why we chosen this method: We have trying to use the GPU-hardware accelerate from CUDA toolkit, python threading library and python multiprocessing library but it does not work because of the incompatibility.

- CUDA (GPU) Incompatibility: The primary libraries used in project are librosa (for MFCC extraction) and scikit-learn (for the MLP classifier) but it does not offer native GPU acceleration for these specific operations via CUDA. It does mainly support PyTorch and TensorFlow.
- Python's Global Interpreter Lock (GIL): GIL is restricting the ability of standard to run on parallelism for CPU-bound.The speed of audio loading and MFCC are not faster when of loading audio files because the main library is loading the audio file as sequential algorithm.
- Python multiprocessing: The joblib library have the ability to computing the multiprocessing task for training the audio file as machine learning type with scikit-learn.

3. Computational Performance Analysis: Execution time and resource utilization were recorded while training in both implementations by using **time.time() and psutil**

    3.1. Feature Extraction Comparison: This stage is the most significant performance difference:

    - **First Version Time: 615.14 seconds**
    - **Enhanced Version Time: 115.40 seconds**

    The enhanced version achieved an approximate **5.3x speedup** for the feature extraction process by effectively utilizing multiple CPU cores to process files concurrently.

    3.2. Overall Execution Time Comparison:

    - **First Version Total Time: 642.60 seconds (~10.7 minutes)**
    - **Enhanced Version Total Time: 147.65 seconds (~2.5 minutes)**

      The parallel version was completed the entire process approximately 4.35 times faster than the first version because of the computational bottleneck.

    3.3. Resource Usage (CPU & Memory): The enhanced version was significantly reduced the wall-clock time and its peak memory usage was slightly higher (Final RSS: ~3632 MB vs. ~3146 MB because of managing concurrent processes and data for better performance.

4. Model Accuracy Analysis: The MLP model was training by the dataset and the results is test set using the results from the latest runs.

    - Standard Version Accuracy: 47.14%
    - Parallel Version Accuracy: 46.56%

The results confirm that the parallelization technique applied to feature extraction did **not significantly impact the final predictive accuracy of the model.** The changes difference shows that the accuracy is less than 0.6% is with this type of model training.

- Neutral emotion achieved the highest F1-score (0.55), suggesting it was the most distinguishable class for the model.
- Sad emotion is getting the lowest F1-score (0.32) shows that it is greater difficulty in its classification.
- Other classes (Angry, Frustrated, Happy) showed moderate F1-scores (0.43-0.47).

The overall weighted average F1-score was 0.46-0.47 and it is moderate success in this multi-class classification task with these features and architecture.

5. Conclusion

A machine learning from Thai speech emotion recognition was successfully implemented using MFCC features and an MLP classifier with the test accuracy of approximately 47%. The sequential feature extraction process is doing a bottleneck computation. By implementing parallel feature extraction using joblib. The overall execution time was reduced by approximately 77% (from ~10.7 minutes to ~2.5 minutes) without affect with the model's predictive accuracy. This can help us to minimize the time consume by using the parallelization for improving the efficiency of computationally in machine learning of audio analysis and It can save more time on large datasets. While achieved accuracy on the complexity of the task. In the future work we could explore alternative feature sets and trying different model architectures like CNNs, LSTMs and hyperparameter optimization to potentially enhance classification for better performance.

6. GitHub repository for source code and performance result: https://github.com/Cloudy-s-OS/Audio-Emotion-Analysis