cogito Intelligence Augmentation (IA) for AI Hackathon

Links:

- Competition Site
- Kaggle Dataset

Folder Structure:

Once the zip file is unzipped following structure can be seen

- site (contains files related to WebApp)
- code (code files)
  - **`mp32wav.py`**: to convert the **`mp3`** files to **`wav`**
  - **`IA for AI.ipynb`**: code for training and testing
- submissions: To scored submissions
  - submission_{score}.csv
- demo.mkv: demo video of WebApp
- Details of my Approcah.pdf: full details of my approach

Problem Statement:

Given a dataset of audio files and the emotion of that audio, Predict the emotion of new audio

- **Type:** Multiclass Classification
- **Metric:** Accuracy
- **Dataset Details:**
  - TrainAudioFiles: mp3, wav training files
  - TestAudioFiles: mp3, wav test files
  - train.csv: labels for training data

- test.csv: names of test files
- sample_submission.csv: submission file

My Approach:

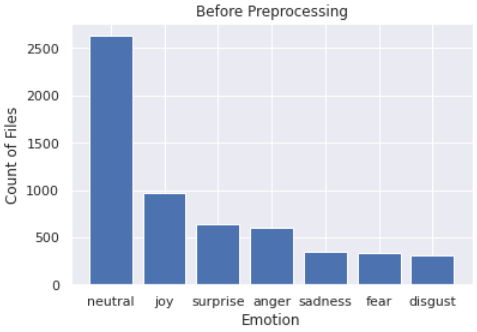
Machine Learning Model

Data Pre-processing:

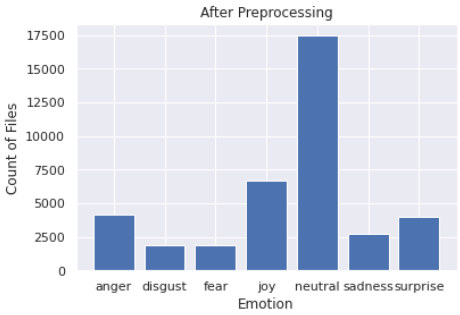
- all mp3 files are converted to wav files using `mp32wav.py` code.
  - Libraries used: pydub
- Created embeddings(openl3) for both training and test data using pretrained audio models.
  - Embedding size: 512
  - Libraries used: openl3, soundfile
  - There are different arguments present in openl3 which I changed to obtain different embeddings.
  - Best embeddings are when [**`input_repr="m11256", hop_size=0.5, content_type="env"`**]
- For a given audio file embeddings are of shape `(N, 512).N` depends on the duration of audio.
- I converted each embedding to `N 512D` embeddings and given same label to all those which I later used for training.

Data Visualization:

Before Pre-processing



After Pre-processing



Training:

- KNN Classifier with standard-scaler is used to train embeddings
  - Libraires used: sklearn
  - **`K Fold cross validation`** with different scaling methods, splits are also experimented.
  - I experimented with different **`CNN`** models, created embeddings from pretrained models like **`VGGish`**, **`edgel3`**. But **`openl3`** with KNN Classifier gave good results.

Testing:

- Finally, the model is saved and used for testing.
  - In testing embeddings are created and then split in same way as in training. Predictions are made for each 512D embedding and scores are averaged over all those to get final prediction.

Application

How to use:

- Create a new **`conda`** environment (preferable)
- Open anaconda prompt and activate above environment
- Got "**`site folder`**"
- Install requirements (pip install requirements.txt)
- Run app.py file, it will give a local hosting link **(http://127.0.0.1:5000/)**.
- Open it and upload audio file, you can see the wave form of audio appears on right when audio is playing.
- Click on "predict" button. On left-top predicted emotion is displayed. On left-bottom predicted probabilities of each emotion are displayed.

Tech stack used: HTML, CSS, JS (Frontend), Flask (Backend)

Highest score: 58.87663

Embeddings from openl3, KNN Classifier with k = 3, weights="distance" and standard-scaler scaling before KNN

Future Scope

- We can train the audio data with CNNs, pretrained audio models.
- Model size can be optimized as well as speed using **GPU** and **RAPIDS** library
- A recording audio option can be added to WebApp through which we can directly predict the emotion by recording real time audio.