Detecting Depression through Patient Speech

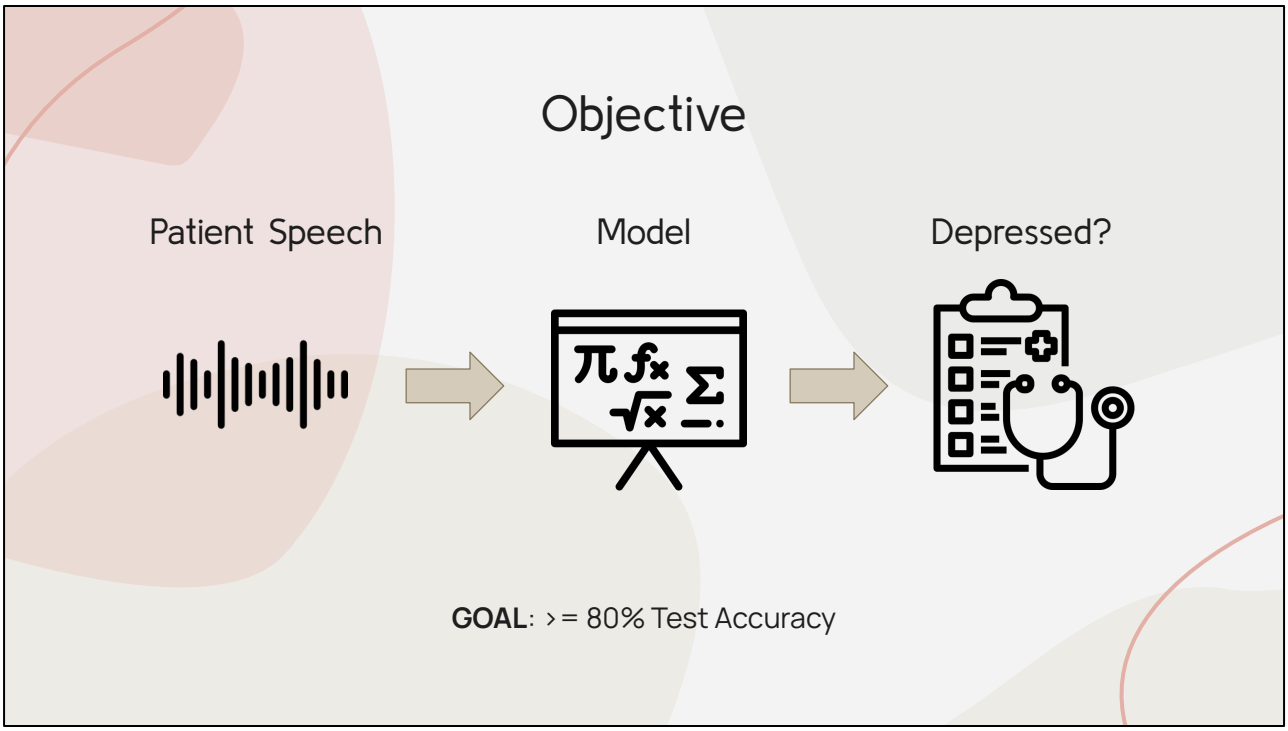
Team: Brian Lin

Hi Im brian and my project is using patient voice recordings in order to predict depression. I'll do a quick recap of what I presented for the midterm presentation, before going more in depth about the modelling, results, limitations, and future work.

Problem Statement

- Current standard for diagnosis are self reported questionnaires
- Several Limitations:
  - Patient not speaking up
  - Dependent on patient mood
  - Differing perceptions of depressive feelings
- More reliable diagnosis method can provide earlier intervention, improving patient outcomes and lowering costs
- 2/3 of depression cases in the US go undiagnosed

The current standard for detecting depression in patients is through self reported questionnaires. These are often prone to error, due to many reasons. By having a more objective method to diagnose depression, we can help to mitigate this issue. There have been several studies that have shown that pitch, tone, and other acoustic features can be used to help detect depression, so I want to build on top of these studies by exploring different types of models and features to tackle this problem.



The goal of this project is to build different machine learning models that take in audio clips of patient speech as its input, and output whether the patient is depressed or not depressed. Success for this project would be creating a model that achieves >=80% accuracy on the test set, while failure would be creating models that achieve <80% accuracy on the test set.

Dataset

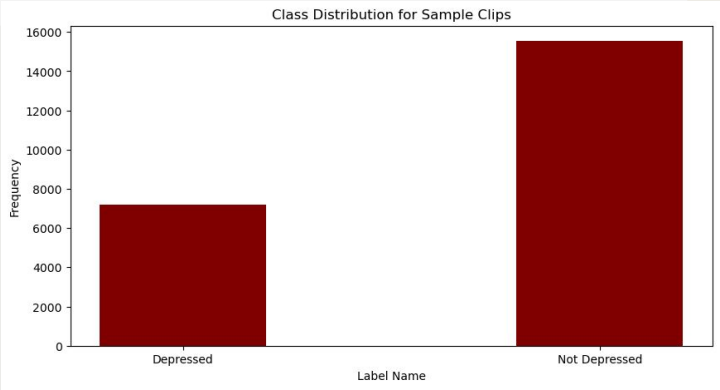
- DAIC-WAZ Dataset from USC Institute of Creative Technology
- Contains audio data + output labels of “depressed” vs “not depressed from 189 interview sessions between participants and virtual interviewer
- Total of ~50.4 hours of audio data
- 80-10-10 split

Figure 1: Ex. Virtual Interview

The dataset comes from USC's Institute of Creative Technologie sand consists of 189 sessions, averaging 16 minutes, between a participant and virtual interviewer called Ellie, controlled by a human interviewer in another room via a "Wizard of Oz" approach. Prior to the interview, each participant completed a psychiatric questionnaire (PHQ-8), from which a binary "truth" classification (depressed, not depressed) was derived. Also worth mentioning that I ended up Employed a 80-10-10 split on the dataset

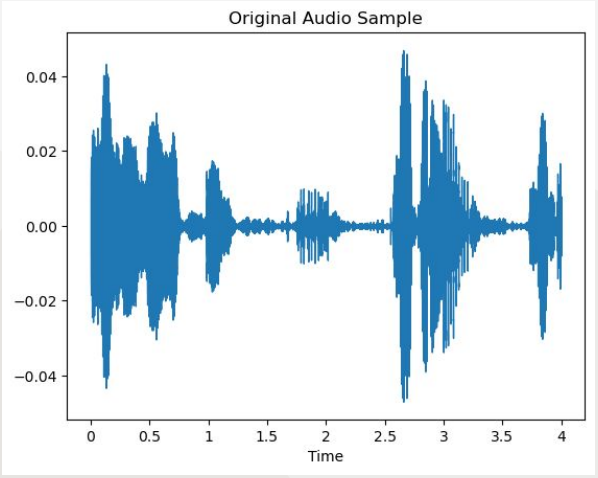
Data Preprocessing: Cleaning and Splicing

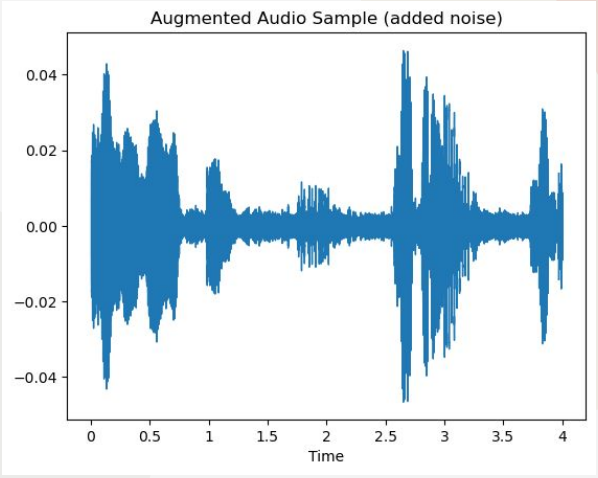
- Remove silence and virtual interviewer portions from audio
- Splice audio into 4 second clips
- 22,730 total samples after cleaning



First there some data cleaning to be done. The main part was to removed the silence and the virtual interviewer portions from the audio, since we only care about the participant audio portions. Then I split the audio into 4 second clips to standardize the length of the samples.

Data Preprocessing: Adding Augmented Samples

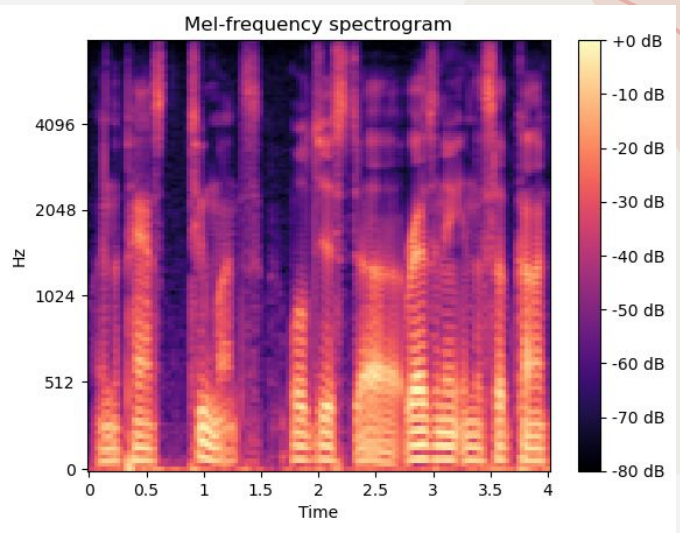




I also added more training rows by augmenting the audio samples. I doubled the training set by adding an augmented audio sample that contained some random noise. Doing this simulates the audio having a bit more background noise, so it shouldn't have any effect on the potentially helpful signals for depression.

Feature Engineering 1: Mel Spectrogram

- Converted audio data into image based data (spectrogram) to run CNN on
- Visually represent a signal's loudness, or amplitude, as it varies over time at different frequencies
- Converted frequencies to Mel scale



Previous studies found success using mel spectrograms and CNNs, so I wanted to start with this model type. CNNs require a visual image I transformed the audio using a spectrogram. A spectrogram is a visual representation of sound, displaying the amplitude of the frequency components of a signal over time. Specifically, the type of spectrogram I used is a Mel-frequency spectogram which scales the frequencies in a way more similar to how humans perceive frequencies.

However, after building a couple of models using mel spectrograms as the input, it did not seem to work well for this specific task, so I decided to try using a different feature set. The openSMILE package was chosen due to its previous success in feature extraction for audio classification tasks. I extracted 25 different low level audio descriptors over time, sampled every 0.01 seconds, and used them as the input into my model. Low level audio descriptors look to capture various aspects of audio signals, from basic energy and loudness to more complex features like MFCCs and spectral characteristics. When flattened, a total of 9900 features were extracted. By leveraging these descriptors, the model aimed to capture a more detailed representation of the audio signals.

Feature Engineering 3: YAMNet Embeddings

- Pre-trained deep neural network designed for audio classification
- Developed by Google Research
- MobileNetV1 architecture
- Extracts 1,024-dimensional embeddings from audio waveforms

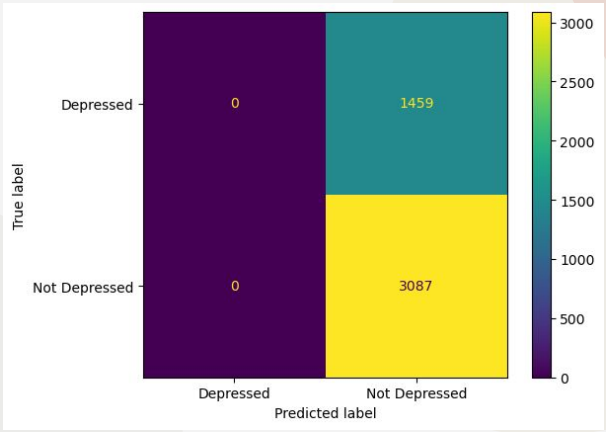
Additionally, I also wanted to experiment with popular pretrained audio classification models, more specifically YAMNet. YAMNet, is a pre-trained deep neural network designed for audio classification, capable of identifying 521 distinct audio event categories from the AudioSet dataset. It was developed by Google Research and employs the MobileNetV1 architecture and extracts 1,024-dimensional embeddings from audio waveforms, which I used for the input for my transfer learning models.

Baseline Model: Majority Class Predictor

```
Train Accuracy: 0.6850711247983575
Validation Accuracy: 0.6790585129784426

AUC Score: 0.5

F1 Score: 0.8088562819337088
Recall: 1.0
```



First, I created a quick majority class predictor just to have some values to compare my later models against. The evaluation metrics that I am using in this project are **Accuracy**, **AUC score**, **F1 Score,** and recall. Recall is important in this case since it Measures the percentage of correctly labeled rows when the true label is “depressed” which is particularly important as depression often goes undiagnosed.

Mel Spectrogram Based Models

	Feature Set	Accuracy	AUC	F1 Score	Recall
Majority Class Predictor	None	0.681	0.500	0.810	1.000
Logistic Reg	Mel Spectrogram	0.663	0.520	0.787	0.910
Random Forest	Mel Spectrogram	0.681	0.500	0.810	1.000
CNN	Mel Spectrogram	0.677	0.560	0.789	0.887

My initial approach utilized Mel spectrograms as features for the classification task. I implemented logistic regression, random forest, and convolutional neural network models, with different hyperparameters and architectures. However, even when tuning the hyperparameters, the performance of these models did not significantly surpass that of the majority class predictor, suggesting that only using the Mel spectrograms as features, may not be providing the necessary information for this task. This observation prompted me to experiment with feature extraction methods or the integration of additional features to improve the model's performance.

openSMILE Based Models

	Feature Set	Accuracy	AUC	F1 Score	Recall
Majority Class Predictor	None	0.681	0.500	0.810	1.000
Logistic Reg	OPENSmile	0.694	0.578	0.819	0.973
Neural Network	OPENSmile	0.701	0.582	0.824	0.971
LSTM	OPENSmile	0.681	0.509	0.807	0.981
XGBoost	OPENSmile	0.757	0.691	0.856	0.972
AdaBoost	OPENSmile	0.683	0.541	0.802	0.971

My next set of models involved the extraction of a time series of low-level audio descriptors using the openSMILE Python package. I implemented logistic regression, neural network, LSTM, XGBoost, and AdaBoost models, with different hyperparameters and architectures using this feature set. This feature set seemed to be slightly better suited for this task. Despite a majority of the models still not being able to achieve an accuracy much better than the majority baseline predictor, the models appeared to be learning some information that is useful for predicting depression.

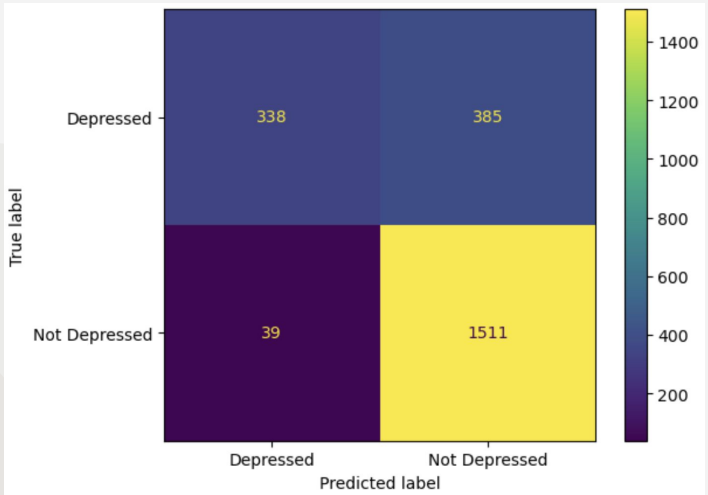
A common trend observed across models was the tendency to predict the majority class for nearly all samples. This behavior suggests that the models may not have been sufficiently trained or the features selected did not provide sufficient information to separate the classes. However, the openSMILE based models did this to a lesser degree, as seen by the slightly higher AUC scores from these models, while still being able to keep a higher accuracy.

A couple of interesting results worth pointing out. XGBoost performed well while AdaBoost did not perform well. AdaBoost has been noted that it may not perform as well when faced with irrelevant features. This limitation could explain why AdaBoost did not achieve the same level of accuracy as XGBoost.

The performance of LSTM and other neural networks was surprisingly underwhelming for this task. Despite the effectiveness of LSTMs in handling sequential data, these models did not yield great results. This outcome may be attributed to the data used or

it just a need for further tuning and experimentation with the model architecture and feature selection.

XGBoost Using openSMILE Features

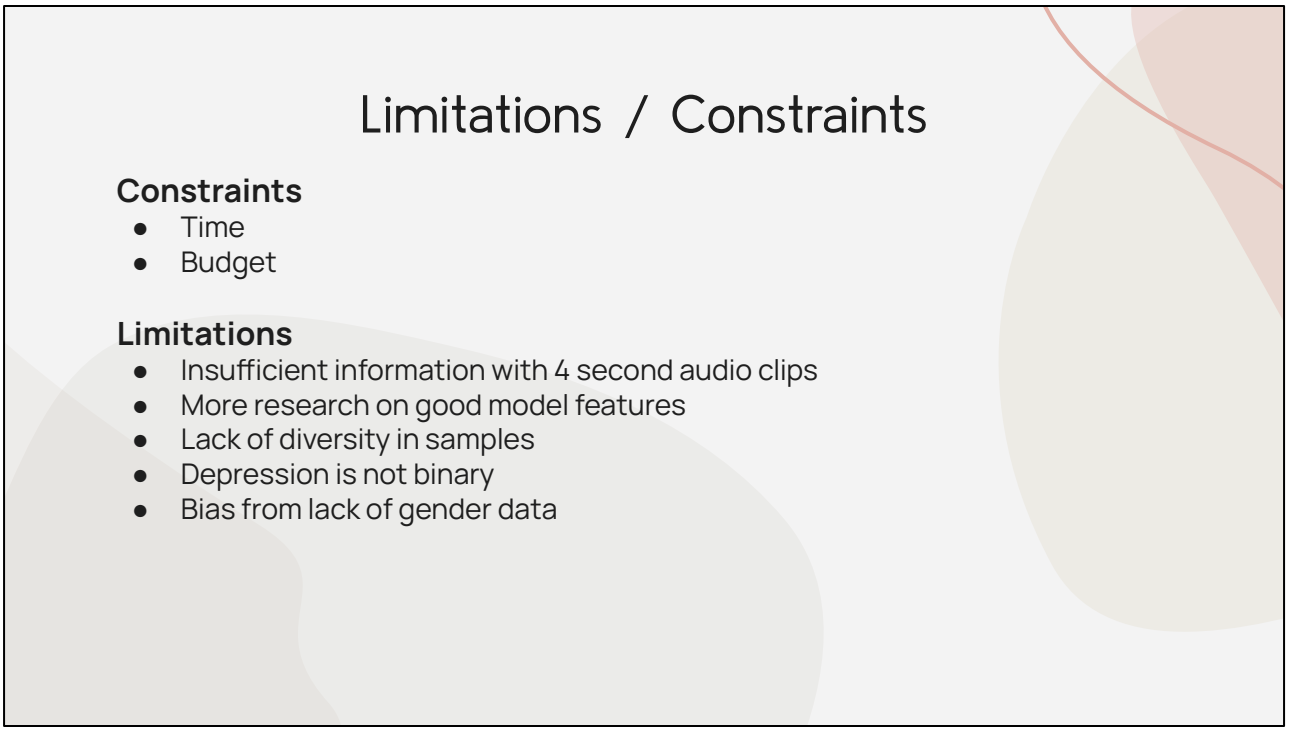


What was particularly encouraging about the potential of the openSMILE feature set for this task was the performance of XGboost. Although still far from perfect, the XGboost model performed much better when using this feature set, with high accuracy, AUC, F1, and recall. Additionally, when looking at the confusion matrix for the model below, we can see that the model is doing very well when the true class is “Not Depressed”. When the true label is “Depressed”, the model has a harder time but is still a lot more successful than previous models. These results support that using the time series of low level descriptors extracted by openSMILE works better for this task compared only with a mel spectrogram.

Transfer Learning Models

	Feature Set	Accuracy	AUC	F1 Score	Recall
Majority Class Predictor	None	0.681	0.500	0.810	1.000
YAMNet + NN	YAMNet Embeddings	0.681	0.500	0.810	1.000
YAMNet + XGboost	YAMNet Embeddings	0.685	0.508	0.811	0.995

Next, the YAMNet model was explored as a potential tool for transfer learning. The exploration of using XGBoost yielded promising results when applied to low-level descriptors. However, the transition to using YAMNet embeddings directly with XGBoost did not yield the similar success. Neural networks using the YAMNet embeddings also did not perform well. Similar to the mel spectrograms based models, the model performed best when predicting the majority class for all or most samples, which leads me to believe that embeddings may not be providing the useful information to the model for this task. YAMNet was originally trained to identify audio events such as a dog barking or like footsteps. It is likely that the task of identifying depression from someone's voice is looking for much different signals than the original events, which is why these models did not perform well.



There are several reasons that I believe contribute to the poor performance of the models. One potential reason is the insufficient information provided by 4-second audio clips as inputs to the model. It is possible that depression detection requires a more comprehensive analysis than what can be captured in such short segments. With more time, I would have wanted to experiment with different audio sample lengths as inputs.

Another possibility is that the chosen features may not have been optimal for the task at hand. Depression detection through patient speech is a distinct task that differs significantly from more common audio classification tasks such as speaker or language recognition. Exploration of specific features that are known to contribute to detecting depression, rather than relying on features that are generally beneficial for a broader range of audio classification tasks would have been good.

This project was unable to take advantage of findings from previous studies. Numerous successful projects have been commercialized, and the specific audio features or model types that were found to be connected to depression were not publicized. If the project had a larger budget, market research would have been a significant area of focus.

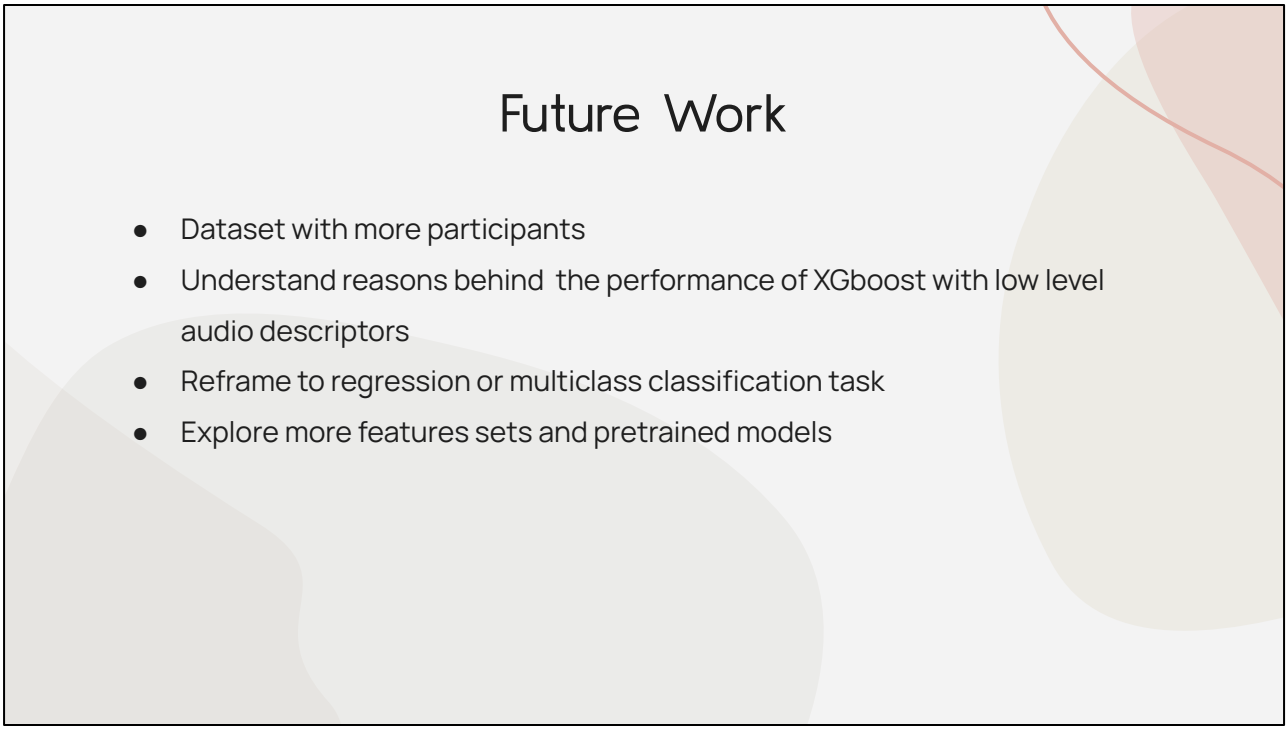
The diversity of the dataset used for training the model is another factor that could impact its performance. With only 189 unique participants contributing to the 22,730 original samples, there is a risk that the model may not be able to learn effectively from such a limited set of participants. It is likely that the model has not encountered

enough variation in the training data to develop a robust understanding of the patterns associated with depression.

Additionally, the study does not acknowledge the complexity of depression, which exists on a spectrum rather than being a binary condition.

Lastly, the absence of gender data in the model is another potential limitation. Gender plays a significant role in the manifestation and treatment of depression, and its exclusion from the study may lead to biased predictions.

These limitations highlight the need for future research to address these issues.



Although this initial study did not produce the most promising results, I believe there is still a lot of potential in future work. The first step would be to collect a more extensive dataset, namely with a greater number of participants.

Additionally, investigating and understanding the specific reasons for why XGBoost with the low level descriptors was much better compared to other models in this context could uncover key insights about this task.

Next, by recognizing that depression exists on a spectrum rather than being a binary condition, future research could benefit from reframing the classification task. Instead of a binary classification approach, a regression or multiclass classification task could be more appropriate.

Lastly, with more time I would have wanted to explore transfer learning with other pretrained models.

Standards

- **Language**: Python 3.11
- **Packages**: Pandas, Numpy, Tensorflow, Tensorflow Hub, XGboost, Scikit Learn, Librosa, PyAudioAnalysis, openSMILE
- **Processor**: M2 chip
- **Development environment**: Local on 2023 macbook air
- **IDE**: VScode

The primary programming language used in this study was Python. The main python packages used for the model building process were Pandas, Numpy, and Tensorflow. TensorFlow Hub was used to access pre-trained models. The XGBoost library and Scikit-learn was used for accessing boosting algorithms. For audio processing and feature extraction, Librosa, PyAudioAnalysis, and OpenSMILE was used.

The development environment was set up on a locally-hosted M2 2023 MacBook Air, with Visual Studio Code as the IDE and Python 3.11 as the programming language.

Thanks for Listening!