Speech Emotion Recognition

CS 4375.001 Machine Learning Final Project Report

Yiran Li and Matt Limsiaco

Introduction

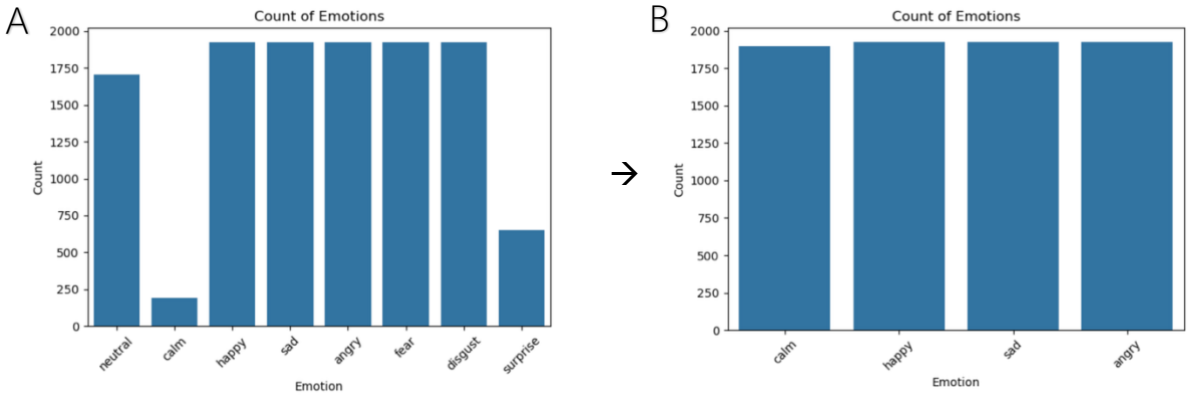
Emotion recognition from speech has become an increasingly important area in human-computer interaction. It enables more natural and adaptive communication between users and AI systems. By automatically detecting emotional states from voice, applications ranging from virtual assistants to mental state monitoring systems can recognize and respond more appropriately to users' needs. Our objective for this project is to detect and classify emotions in human speech by utilizing a One vs. Rest ensemble method to categorize emotion based on audio signal of different emotions.

Datasets and Processing

We found 4 widely used emotional speech datasets online from various sources and combined them together:

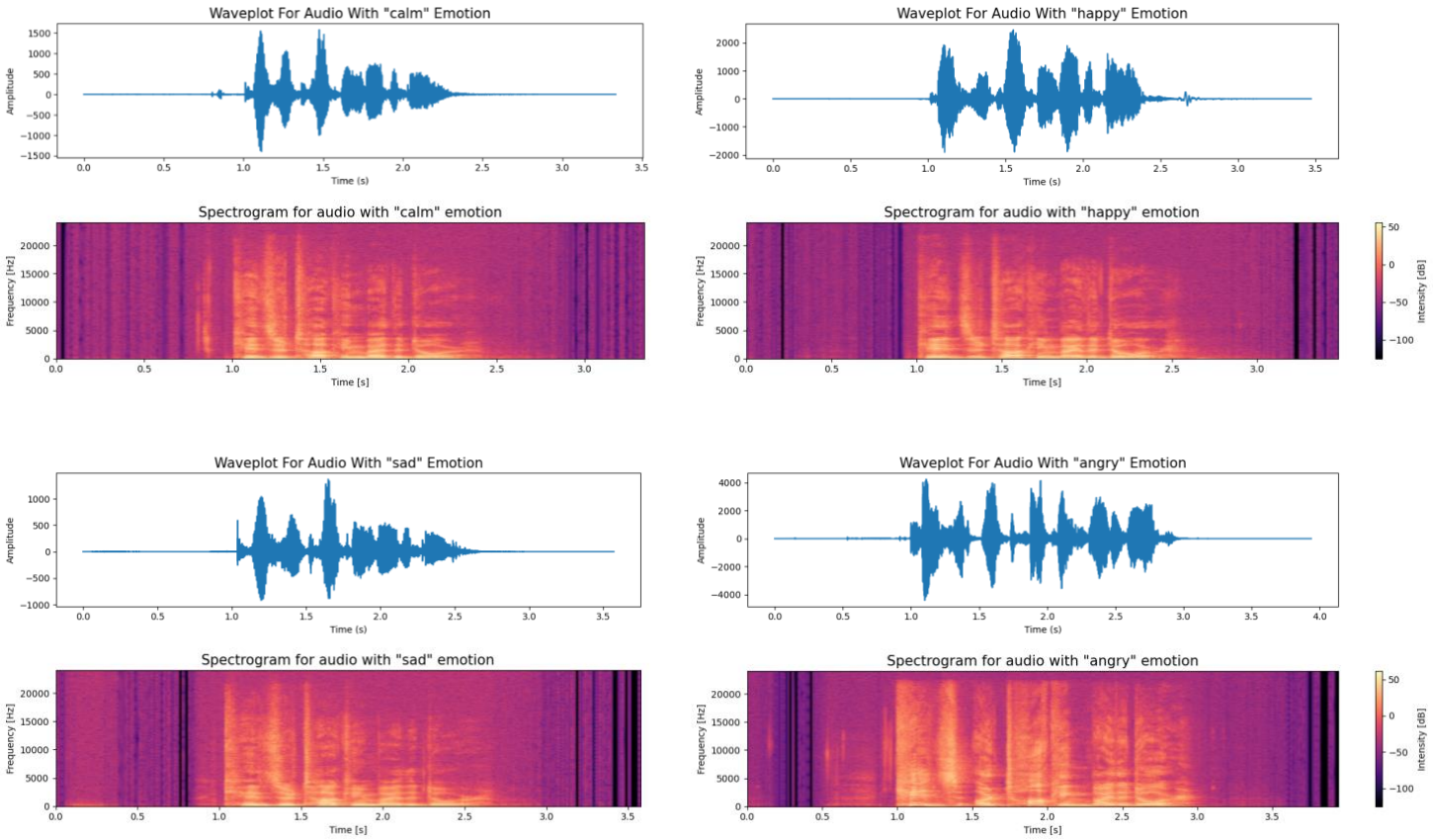
- CREMA-D (Crowd-sourced Emotional Multimodal Actors Dataset)
- RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song)
- SAVEE (Surrey Audio-Visual Expressed Emotion)
- TESS (Toronto Emotional Speech Set)

These datasets contain nearly 14,000 speech clips from 121 professional actors and actresses, and the emotions included were happy, sad, angry, calm, neutral, disgusted, fearful, and surprised. For our project, we are only extracting four emotions: happy, sad, angry, and calm. We combined neutral and calm together and created a balanced distribution of examples in each emotion.



(A) Distribution of original combined dataset vs. (B) Distribution after emotion processing

We also created waveplots and spectrograms of each emotion to see the amplitude and frequency spectrum over time.



Feature extraction and augmentations

It's difficult to analyze audio clips because it doesn't give us numbers to analyze right away. Therefore, we need to extract features for the model to understand. The focus is on audio signals which includes time, amplitude, and frequency. We extracted a set of features using the Librosa library. These are commonly used features in speech recognition models that reflect the temporal and spectral characteristics of speech signals. Audio was processed using a frame length of 2048 samples and a hop length of 25% overlap with frame length, which balances time resolution and frequency detail. For feature extraction, we referred to the following papers:

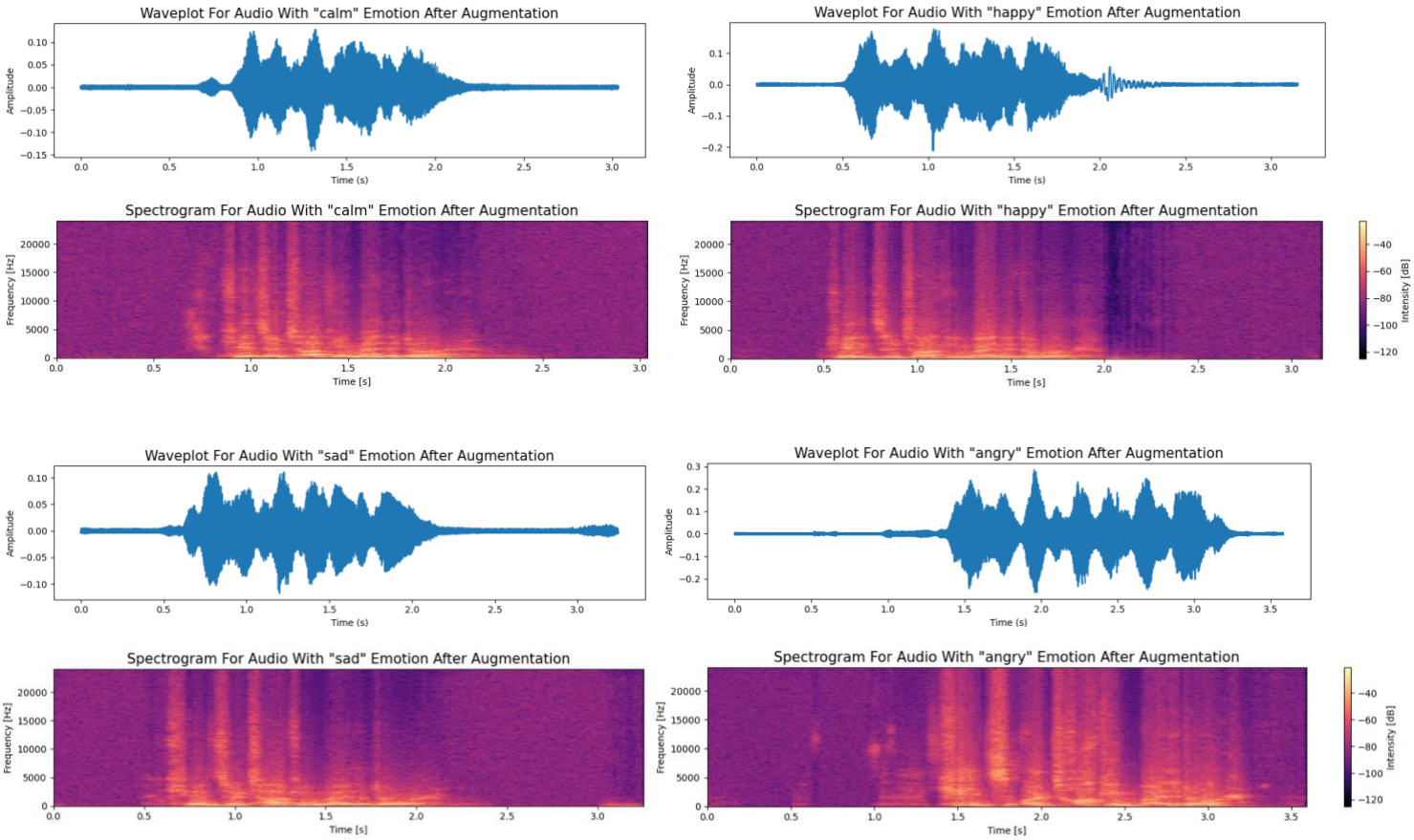
https://medium.com/heuristics/audio-signal-feature-extraction-and-clustering-935319d2225

https://www.researchgate.net/publication/365515286_Data_Augmentation_and_Deep_Learning_Methods_in_Sound_Classification_A_Systematic_Review

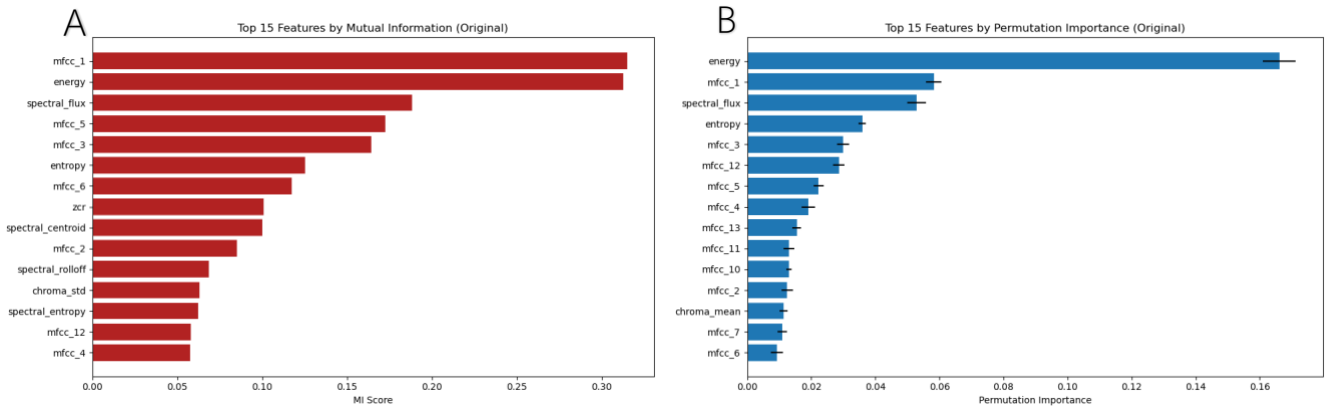
The features are:

- Zero Crossing Rate- Rate of sign changes in the signal wave within a frame; higher values often indicate noisier or more energetic speech.
- Energy- Represents loudness or intensity.
- Entropy of Energy- Measures abrupt changes in energy.
- Spectral Centroid- "center of gravity" of spectrum; higher values mean higher pitched sound.
- Spectral Spread- Variance of frequencies around centroid.
- Spectral Entropy- Entropy of the normalized spectral energies for a set of sub-frames.
- Spectral Flux- Rate of spectral change over time.
- MFCCs (Mel-Frequency Cepstral Coefficients)- commonly used to capture timbral aspects of audio signal.
- Chroma Vector- 12-dimensional representation of spectral energy distributed over the 12 pitches (harmonic and pitch content).

Augmentations on all the features are typical when dealing with audio data, meant to simulate real world variability such as speaking rates, amplitude variations, environmental noises, and slight misalignments in timing. We utilized time stretching, pitch shifting, additive noise, time shifting, and dynamic range compression. Including this step helps reduce variance in the model, encouraging it to focus on meaningful emotional patterns rather than irrelevant variations in the data.



Now that the features are processed, we decided to look at feature importance using mutual information and permutation importance. Mutual information measures how much knowing a particular feature reduces uncertainty about the emotion label, while permutation importance, on the evaluates the drop in model performance when a feature's values are randomly shuffled, providing a model-specific view of how much that feature contributes to predictions. For permutation importance, we used a Gradient Boosting classifier as a proxy since applying it to our OVR stacked ensemble would be computationally intensive. Using both methods offers complementary insights based on information theory and performance:

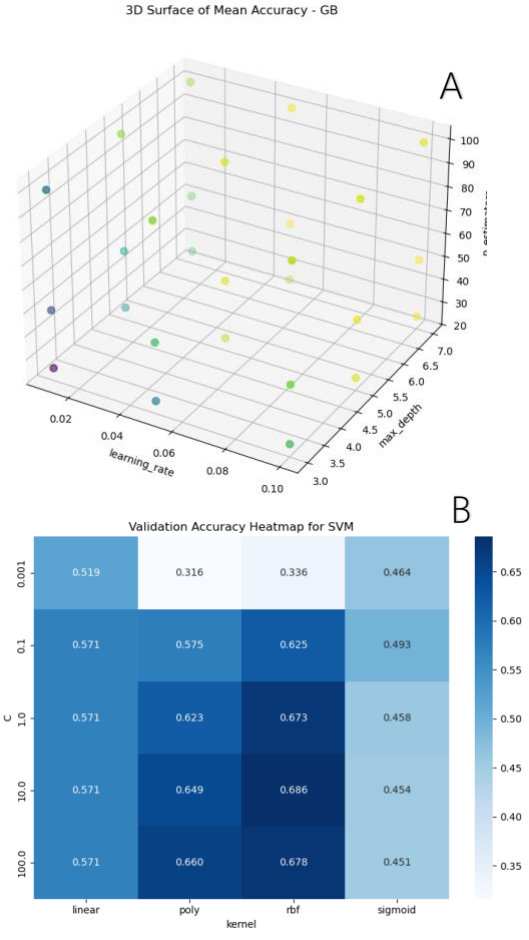


(A) Mutual Information and (B) Permutation Importance feature rankings on original dataset

Hyperparameter Tuning

To start our One vs. Rest stacking ensembling, we found the best tuning hyperparameters for each of our "mini ensembles": logistic regression, SVM, KNN, decision tree, and gradient boosting. This was accomplished by utilizing grid search, training and testing each individual model on all combinations of hyperparameters and comparing the results.

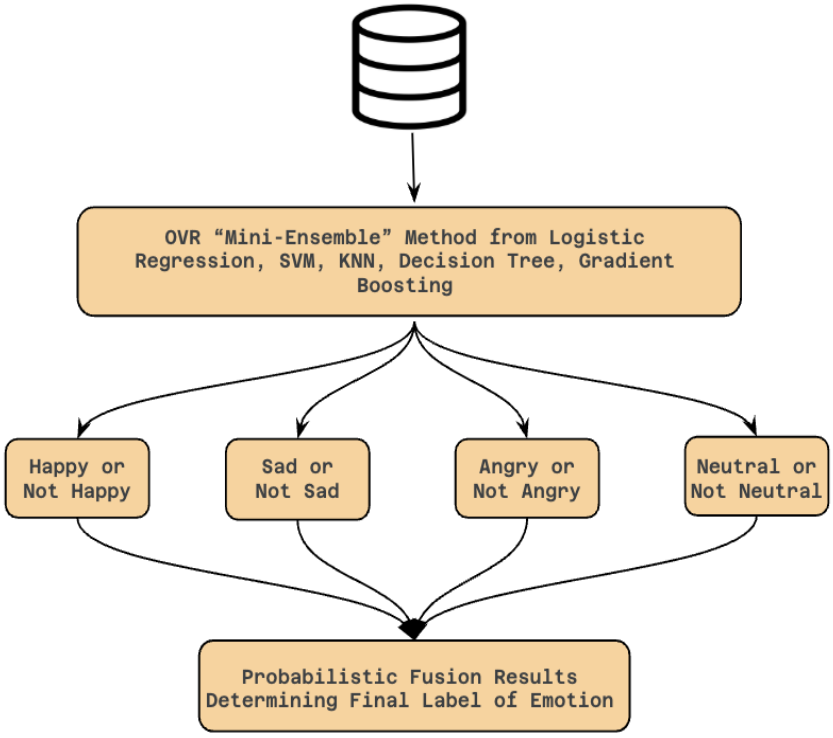
Model	Hyperparameters
Logistic Regression (LR)	`C = 0.1, solver = 'lbfgs'`
Support Vector Machine (SVM)	`C = 10, kernel = 'rbf'`
K-Nearest Neighbors (KNN)	`n_neighbors = 7, weights = 'distance'`
Decision Tree	`max_depth = 10, min_samples_split = 10, min_samples_leaf = 1`
Gradient Boosting (GB)	`learning_rate = 0.05, max_depth = 7, n_estimators = 50`



(A) 3D plot of GB hyperparameter accuracy and (B) Heatmap of SVM validation accuracy.

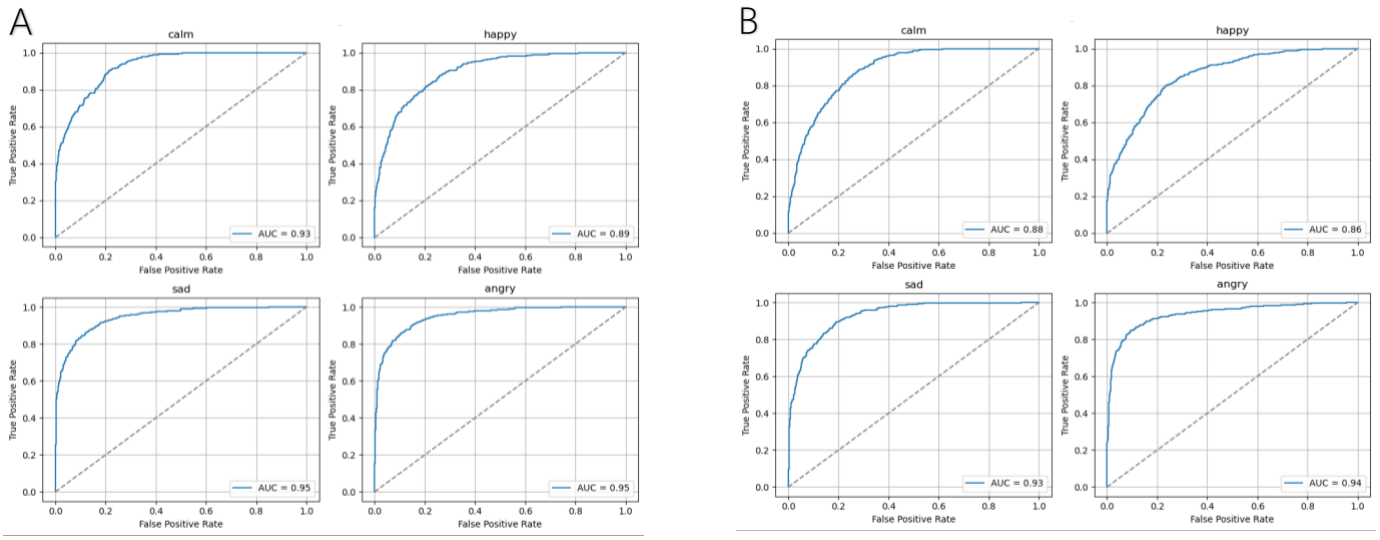
Meta-Learner

We trained a meta-learner model by using the predictions of base models (with their best hyperparameters). Each base model learns from the base set and each base model predicts on the meta set These probabilities will form the meta-features. We chose to train a logistic regression meta-model to combine the base models' outputs and make final decision by assigning different weights to each of their predictions. This ensemble is more flexible and will often outperform the individual models.



Results and Evaluation

For training our stacked ensemble on datasets, we created filtered datasets on the original, augmented, and combined data by using feature importance. Then, we made ROC curves by each of the emotions for all 6 versions of the dataset. Since the ensemble is trained using a one-vs-rest strategy, each emotion is treated as a separate binary classification problem. Therefore, we generated individual ROC curves per emotion to evaluate how well each meta-learner distinguishes its target emotion from the others. This provides a more detailed view of class-specific performance than a single aggregate ROC curve would.



ROC Curves by Emotion of (A) Original Data and (B) Augmented Data

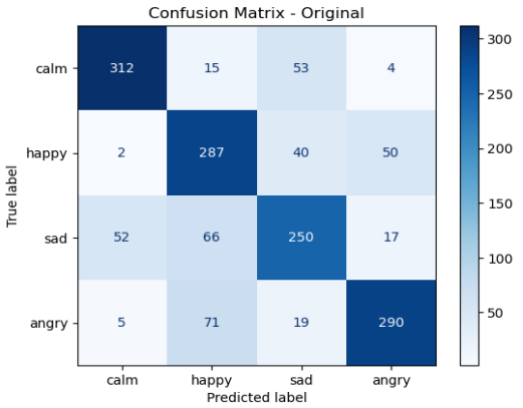
Next, we evaluated predictions on the test set of all datasets. Because we have four emotion classes, we used a 4×4 confusion matrix to visualize model performance across all classes simultaneously. Unlike binary classification, where a 2×2 matrix is sufficient, multiclass classification requires a matrix that shows how often each true class is predicted as each other class. This allows us to assess not only overall accuracy but also which specific emotions are most confused:

```
=== Original ===
Accuracy: 74.30%

Classification Report:
              precision    recall  f1-score   support

       angry     0.8410    0.8125    0.8265       384
        calm     0.6538    0.7573    0.7017       379
       happy     0.6906    0.6494    0.6693       385
         sad     0.8033    0.7532    0.7775       385

    accuracy                         0.7430      1533
   macro avg     0.7472    0.7431    0.7438      1533
weighted avg     0.7475    0.7430    0.7439      1533
```

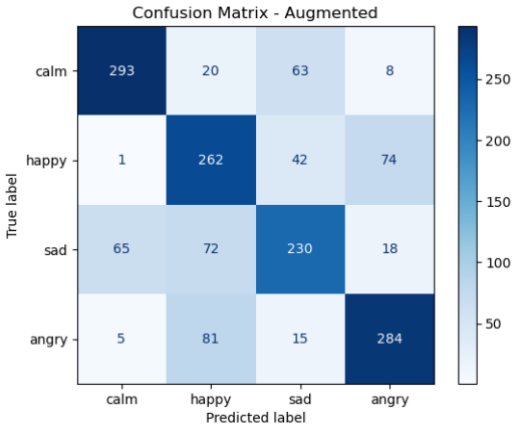



```
=== Augmented ===
Accuracy: 69.73%

Classification Report:
              precision    recall  f1-score   support

       angry     0.8049    0.7630    0.7834       384
        calm     0.6023    0.6913    0.6437       379
       happy     0.6571    0.5974    0.6259       385
         sad     0.7396    0.7377    0.7386       385

    accuracy                         0.6973      1533
   macro avg     0.7010    0.6973    0.6979      1533
weighted avg     0.7013    0.6973    0.6981      1533
```

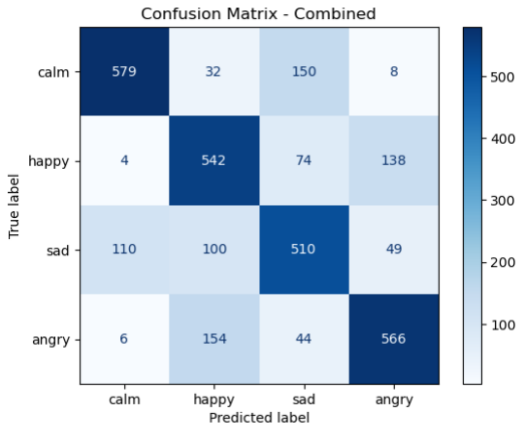



```
=== Combined ===
Accuracy: 71.66%

Classification Report:
              precision    recall  f1-score   support

       angry     0.8283    0.7529    0.7888       769
        calm     0.6546    0.7150    0.6835       758
       happy     0.6555    0.6632    0.6593       769
         sad     0.7438    0.7351    0.7394       770

    accuracy                         0.7166      3066
   macro avg     0.7206    0.7166    0.7178      3066
weighted avg     0.7208    0.7166    0.7179      3066
```

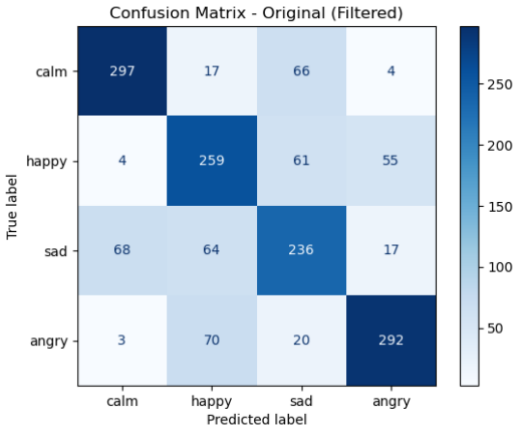


```
=== Original (Filtered) ===
Accuracy: 70.71%

Classification Report:
              precision    recall  f1-score   support

       angry     0.7984    0.7734    0.7857       384
        calm     0.6317    0.6834    0.6565       379
       happy     0.6162    0.6130    0.6146       385
         sad     0.7935    0.7584    0.7756       385

    accuracy                         0.7071      1533
   macro avg     0.7099    0.7071    0.7081      1533
weighted avg     0.7102    0.7071    0.7082      1533
```

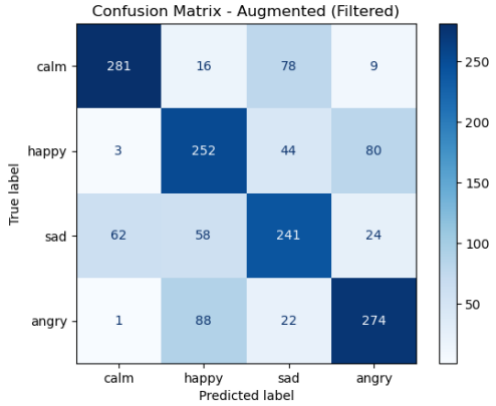



```
=== Augmented (Filtered) ===
Accuracy: 68.36%

Classification Report:
              precision    recall  f1-score   support

       angry     0.8098    0.7318    0.7688       384
        calm     0.6087    0.6649    0.6356       379
       happy     0.6260    0.6260    0.6260       385
         sad     0.7080    0.7117    0.7098       385

    accuracy                         0.6836      1533
   macro avg     0.6881    0.6836    0.6850      1533
weighted avg     0.6884    0.6836    0.6852      1533
```

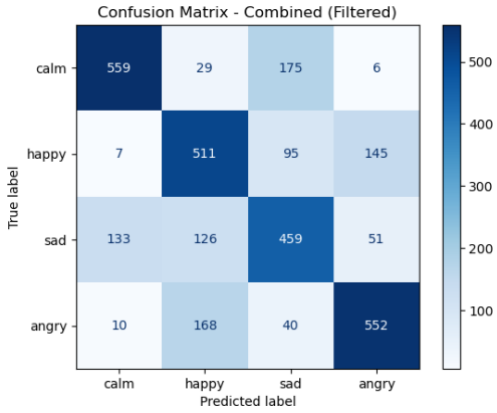



```
=== Combined (Filtered) ===
Accuracy: 67.87%

Classification Report:
              precision    recall  f1-score   support

       angry     0.7884    0.7269    0.7564       769
        calm     0.6127    0.6741    0.6420       758
       happy     0.5969    0.5969    0.5969       769
         sad     0.7321    0.7169    0.7244       770

    accuracy                         0.6787      3066
   macro avg     0.6825    0.6787    0.6799      3066
weighted avg     0.6828    0.6787    0.6801      3066
```



The performance for filtered data has dropped compared to the non-filtered ones. So, we are only fitting CNN on the non-filtered data.

Original data only:

```
Classification Report (CNN Model):
              precision    recall  f1-score   support

       angry     0.6214    0.6667    0.6432       384
        calm     0.4230    0.3984    0.4103       379
       happy     0.4333    0.3714    0.4000       385
         sad     0.6083    0.6857    0.6447       385

    accuracy                         0.5310      1533
   macro avg     0.5215    0.5306    0.5246      1533
weighted avg     0.5218    0.5310    0.5249      1533
```

Augmented data only:

```
Classification Report (CNN Model):
              precision    recall  f1-score   support

       angry     0.6995    0.6667    0.6827       384
        calm     0.4392    0.6675    0.5298       379
       happy     0.5256    0.4000    0.4543       385
         sad     0.6745    0.5221    0.5886       385

    accuracy                         0.5636      1533
   macro avg     0.5847    0.5641    0.5638      1533
weighted avg     0.5852    0.5636    0.5639      1533
```

Combined data:

```
Classification Report (CNN Model):
              precision    recall  f1-score   support

       angry     0.5706    0.6671    0.6151       769
        calm     0.4450    0.5449    0.4899       758
       happy     0.4839    0.2731    0.3491       769
         sad     0.6112    0.6390    0.6248       770

    accuracy                         0.5310      3066
   macro avg     0.5277    0.5310    0.5197      3066
weighted avg     0.5280    0.5310    0.5199      3066
```

Conclusion

The stacked ensemble achieved the highest accuracy of 74.30% on the original (non-filtered) dataset, while the highest accuracy of the CNN model was 56.36% on the combined (non-filtered) dataset. Both models consistently found the "angry" and "sad" classes easier to classify, but the stacked ensemble model was more balanced across all four emotions. This is most likely due to overlapping acoustic features, which make emotions like happy and calm harder to distinguish. The ensemble's superior performance can be attributed to its structure: instead of relying on a single generalist model, it leverages a team of specialized classifiers (one-vs-rest base learners), each optimized for binary classification of a specific emotion. This allows the meta-learner to weigh and integrate each model's strength effectively.

We compared this ensemble approach to a CNN baseline because neural networks are often favored for their ability to learn complex patterns directly from raw or minimally processed data. However, in this example, CNN is expected to underperform due to their reliance on large, high-quality datasets. Speech emotion data, especially when augmented or filtered, introduces variability and noise that deep models struggle to generalize across without overfitting. For filtered data, both models showed weaker performance, which is evident in the reduced test accuracy and ROC AUC values. This decline likely stems from the loss of emotionally informative features, reduced variability, and over-smoothing caused by filtering.

Overall, the results demonstrate that classical ensemble methods with a meta-learner are more reliable and better suited for generalization in noisy, real-world emotion recognition settings than deep learning models like CNNs.

Future Improvements

In the future, we want to focus on expanding the dataset and improving the deep learning pipeline. Additionally, we would explore domain adaptation techniques to reduce performance gaps caused by data filtering and augmentation. We can also implement cross-validation and ensemble averaging across folds to further stabilize model predictions and performance across multiple test sets.

Another key direction is optimizing a new meta-learner for the ensemble. Given that the AUC of each individual binary classifier is already strong, we aim to investigate whether a more advanced or non-linear meta-model can better leverage their strengths and improve final predictions. Lastly, we're interested in systematically studying the scale and type of augmentation that could allow our models to perform reliably on entirely new audio, such as directly recorded voice samples, to make the system more practical and personalized.