# Human Speech Emotion Recognition

The interaction between humans and machines is becoming more and more important as technology progresses. The method in which we communicate and convey messages to computers is every growing. The ability for a machine learning algorithm to detect human emotion will have many applications. This one in particular we are looking at an escalation process in which a call center could detect emotion and increase the quality of its customer service as well as categorize and label call recordings for further analysis.

## Introduction

This following steps focused on collecting data, organizing it, and making sure it's well defined. Attention to these tasks will pay off greatly later on. Some data cleaning can be done at this stage, but it's important not to be overzealous in your cleaning before we've explored the data to better understand it.

## Data Science Problem

The purpose of this data science project is to come up with a predictive model for emotion recognition in speech. We suspects it may have better responses to human / robot interaction based on categories of emotion detect from customers with audio-enabled bots. The categorization of the customer's emotial state can better atune a variety of optional responses to customize interations. This project aims to build a predictive model for emotion recognition based on a set of actor data that falls into 8 categories of emotional state. This model will be used to provide guidance for an audio-enabled bot's available responses and future customer interaction.

## Objectives

We hope to resolve some fundamental questions in this exploration of data.

- Will the provided dataset enable us to successfully detect emotions from human speech?
- There are several emotions to detect; can we identify important ones for this objective?
- Will this project be useful to a call center or other organization?
- Do we have any fundamental issues with the data?

## Literature Review

When researching methods that have been done in the past I came across one that seemed promising. It extracted a few features using the Librosa library and achieved moderate success with four emotions. Through my exploration of the features Librosa was capable of extracting, I wanted to see if we could expand further on the feature set and achieve better results. To accomplish this I found the best available dataset I could and set out to extract the features from it.

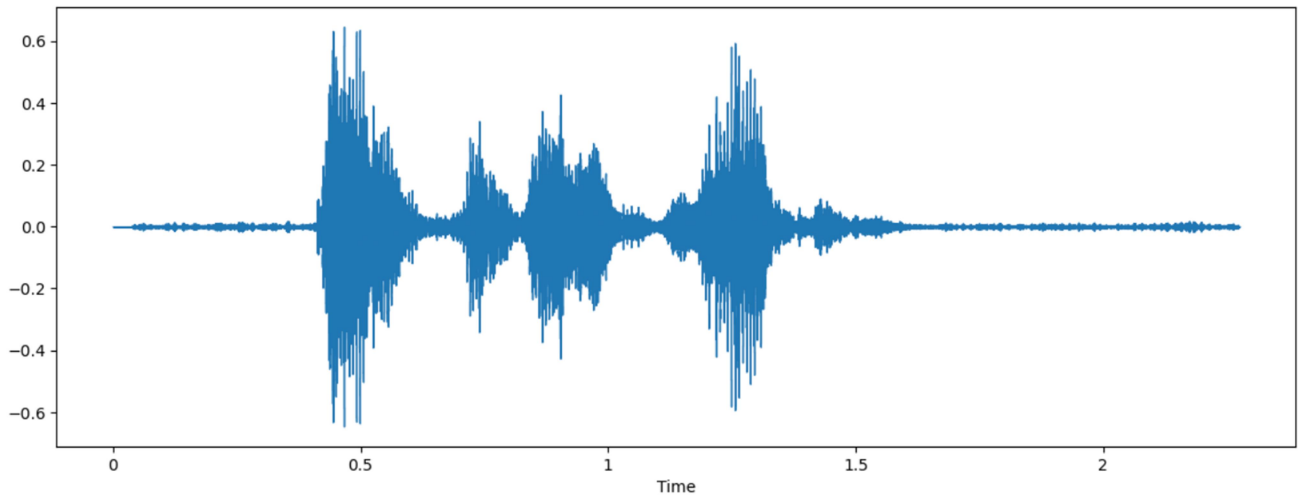
## RAVDESS the dataset

This dataset is pulled from the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). It is a collaboration of professional actors intending to capture emotions in human speech with a neutral North American accent. It contains eight emotions (calm, happy, sad, angry, fearful, neutral, surprise, and disgust) and two intensities (normal, strong). There is also a song version of these but I left this out due to the nature of the objective.

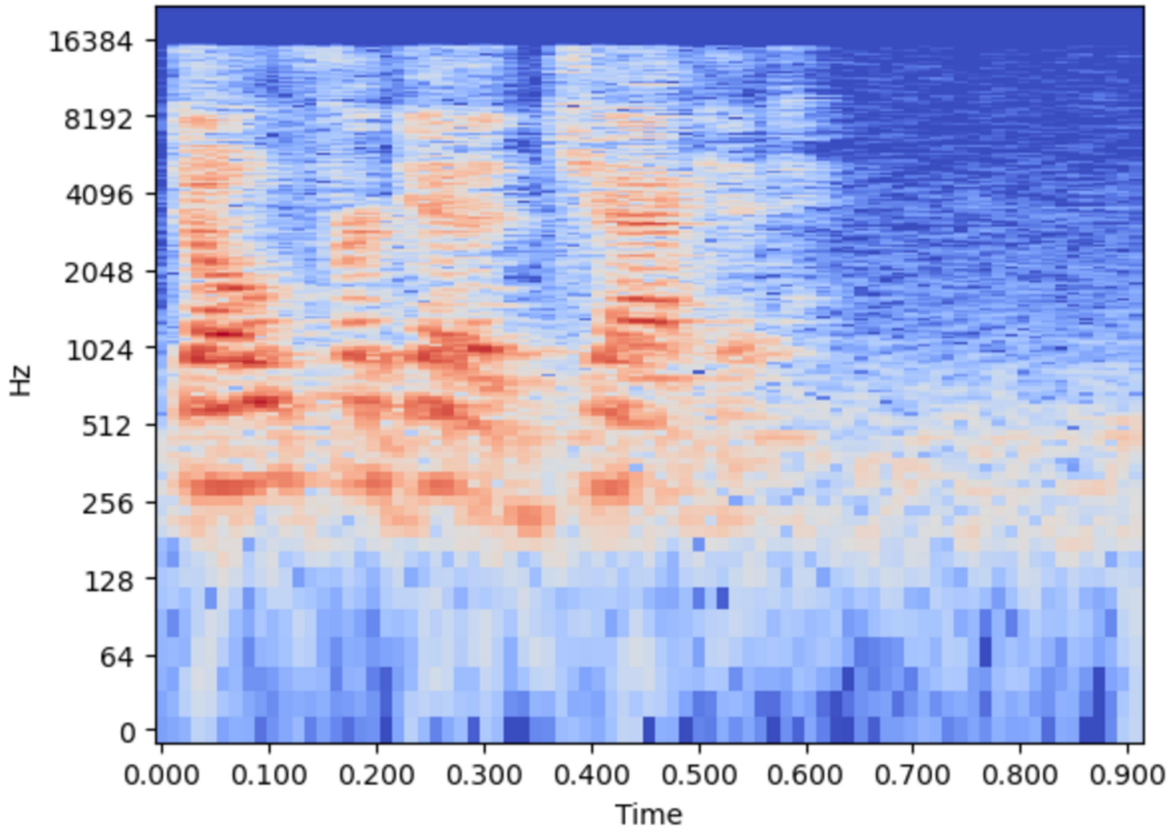
## Importing and Examining

In the upcoming steps I imported the data with Librosa and Soundfile into my workbook and looked at some basic characteristics of the audio files. Here's a wave plot and spectrogram of the first file:

```
<librosa.display.AdaptiveWaveplot at 0x10867749e90>
```



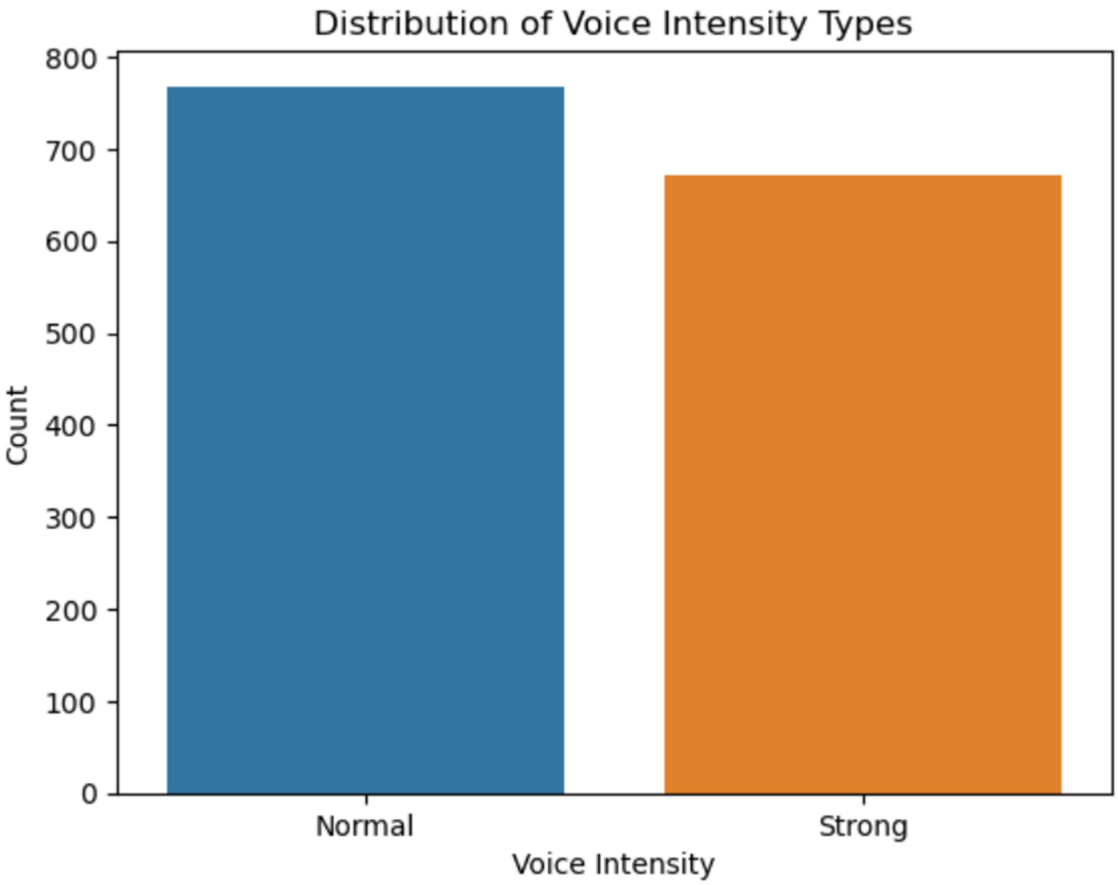
```
<matplotlib.collections.QuadMesh at 0x1adb4854a50>
```

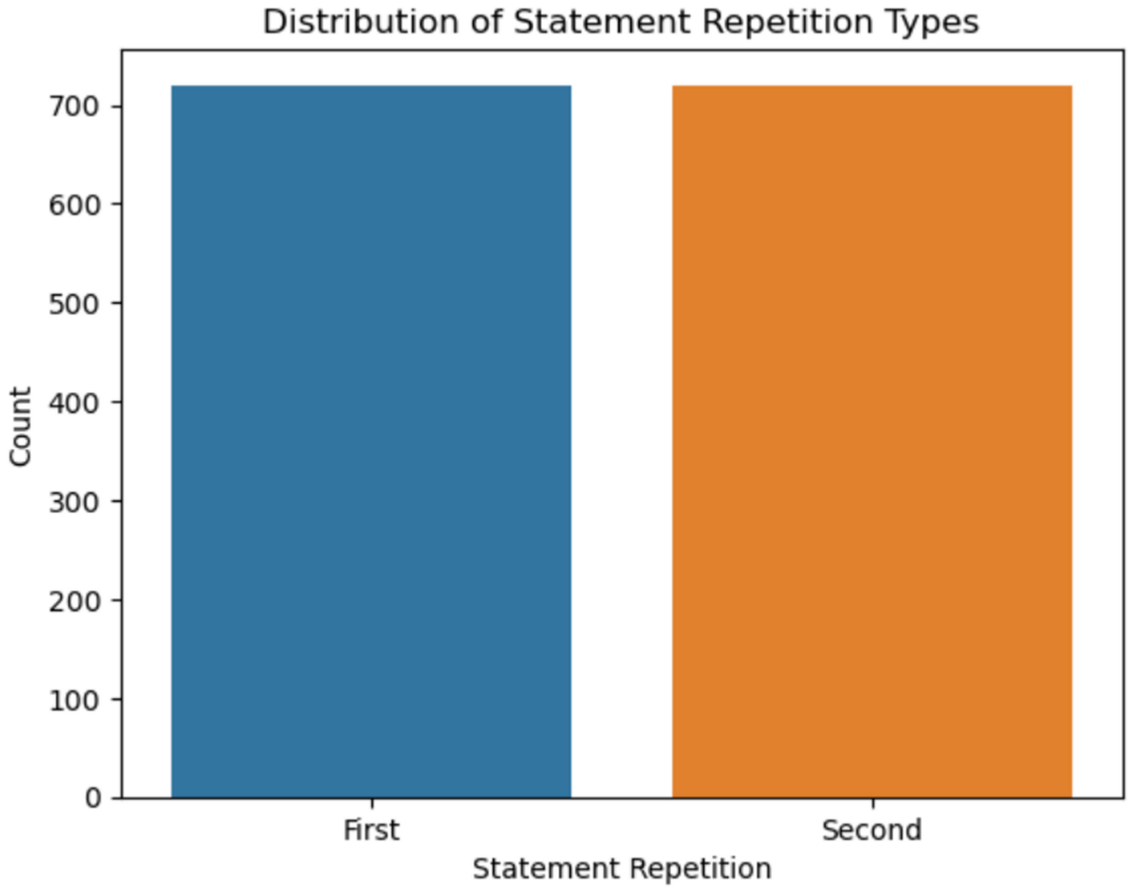


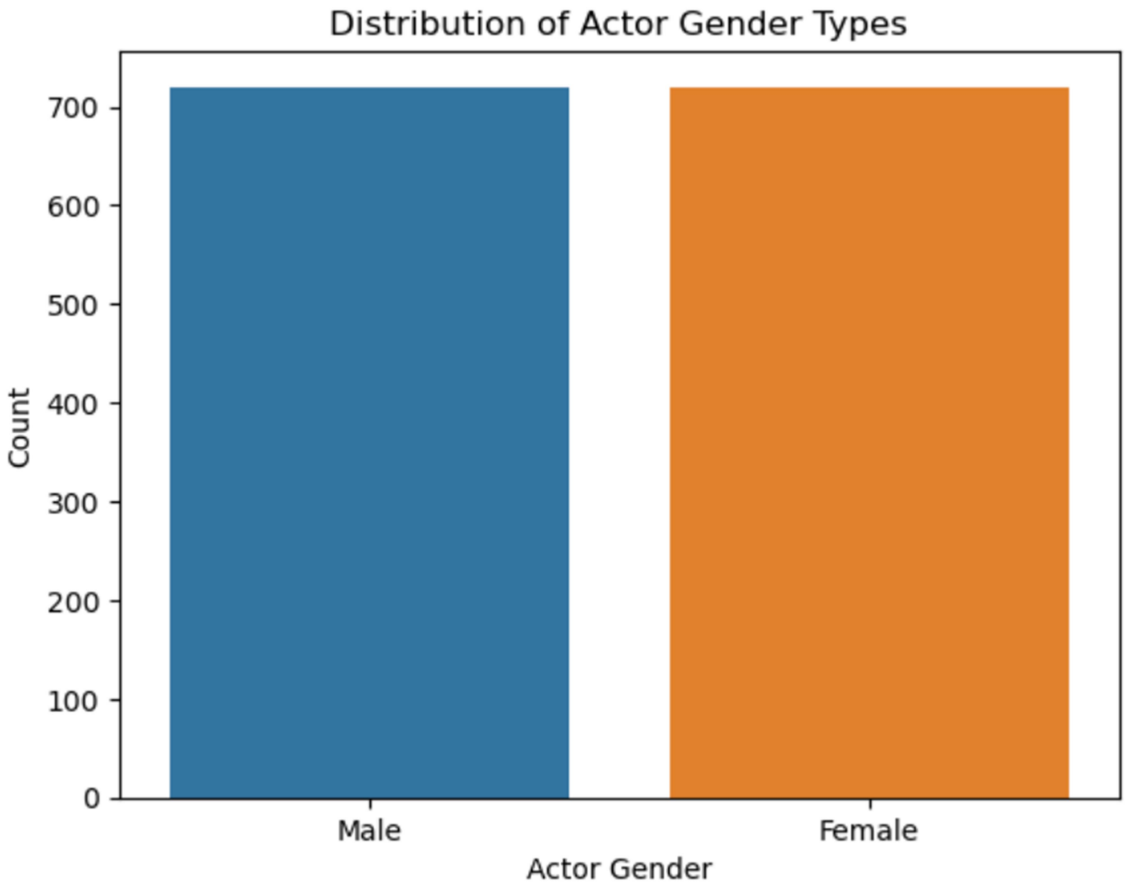

Next I started to explore different features that I could extract and create the dataframe using Pandas. This process was time consuming and I had to go back and redo it several times. Ultimately I ended up with 70 features per audio file and several helper functions.

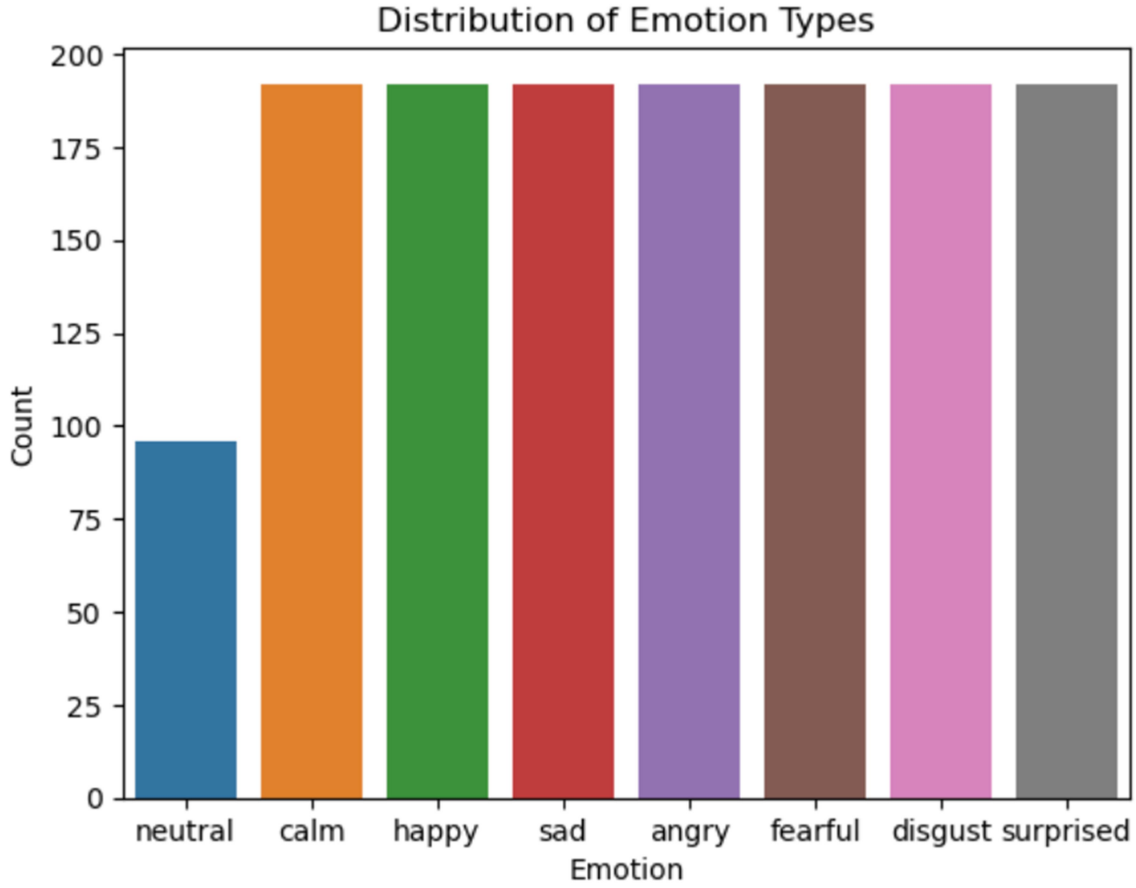
## Exploring

Next I began to explore the dataframe to confirm it's characteristics and usefulness. With the help of Seaborn and Matplotlib I was able to see the distribution of features for some categories:


Distribution of Voice Intensity Types
800
700
600
500
400
300
200
100
0
Count
Normal
Strong
Voice Intensity


Distribution of Statement Repetition Types
700
600
500
400
300
200
100
0
Count
First
Second
Statement Repetition


Distribution of Actor Gender Types
Count
0
100
200
300
400
500
600
700
Male
Female
Actor Gender

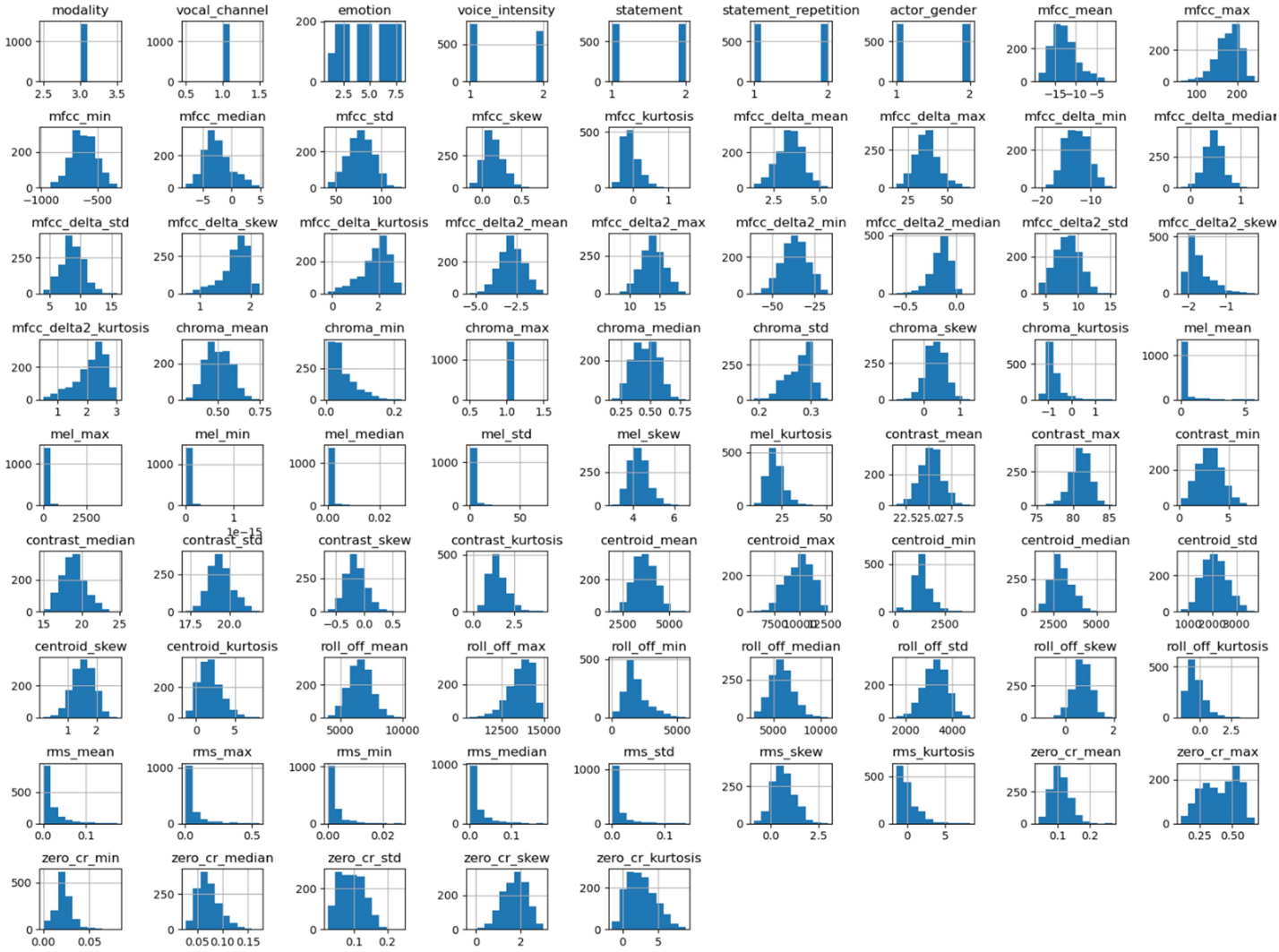

Distribution of Emotion Types
Count
0
25
50
75
100
125
150
175
200
neutral
calm
happy
sad
angry
fearful
disgust
surprised
Emotion

## Cleaning the dataframe

First steps in cleaning the dataframe were to check for duplicates. To do this, I used the '.duplicated()' method to create a new dataframe then checked it's size. Next, I checked for outliers and missing values. The '.isnull()' combined with '.sum()' returned 0 missing values. During the process of feature extraction I came across some that had near infinite values so after adjusting the extraction process I added a check for infinite values as well with '.isinf()' and my final version had none.

## Distribution of Feature Values

In this next section I took a look at some of the values in a broad sense to see if there were any potential problems that could easily be seen.




I also looked at some basic statistics for the feature values with '.describe()'.

## Data Wrangling Summary

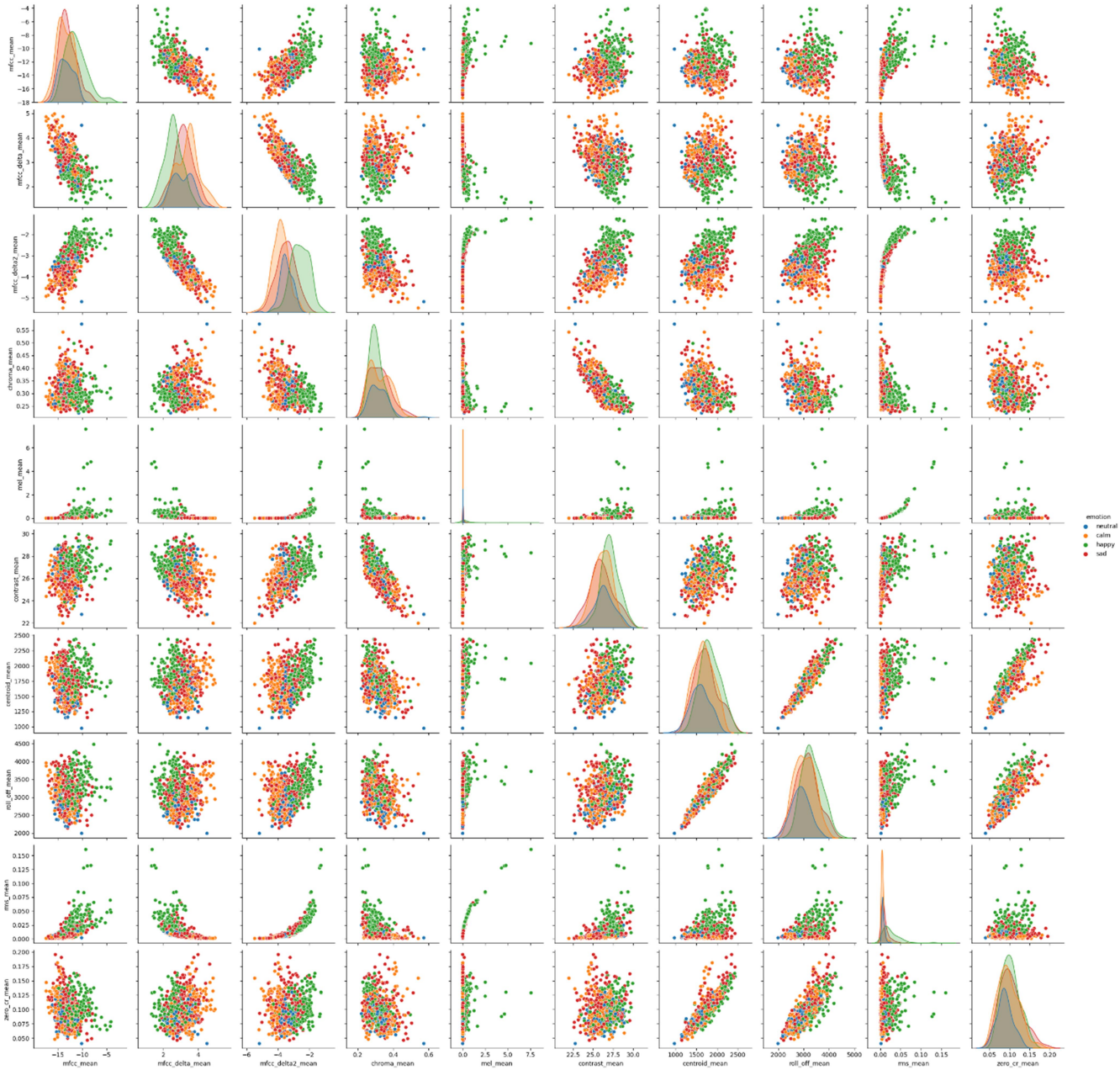
The tests were successful and I was able to successfully import the files into a dataframe. With a little examination of the features I was able to extract we can see an even number of male versus female actors. Same goes for the repetition of statements but the intensity was sqewed but that is understandable since the 'neutral' emotion does not have a 'strong' intensity so there are less of them. I decided not to downsample the audio files upon further research. 16000 Hz seems to be the minimum standard for voice data and I'd prefer not to compromise the later model's ability to accurately predict emotions because of a reduction in quality. Next step is Exploratory Data Analysis and I'm excited to see what patterns arise.

## Exploratory Data Analysis

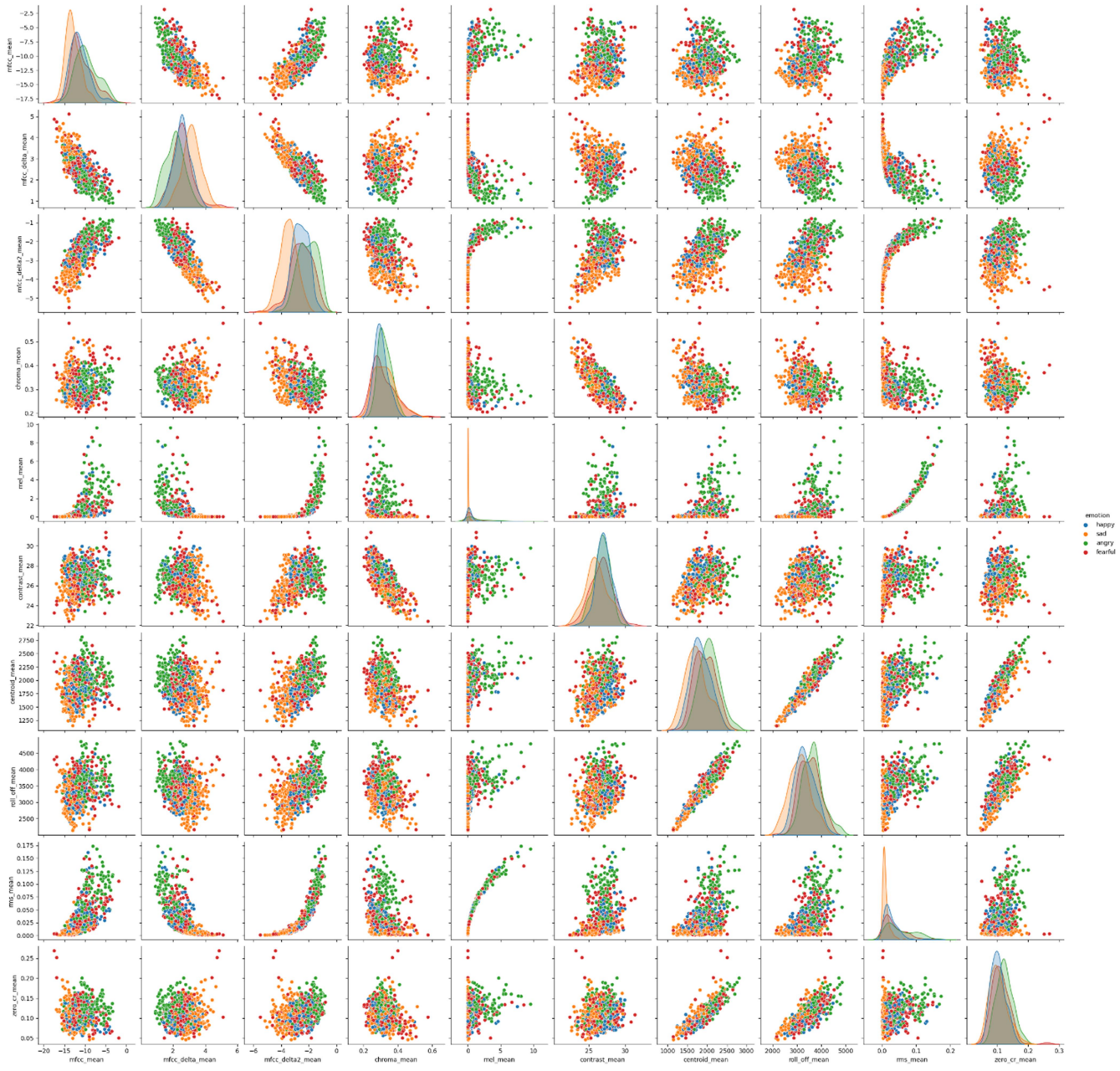
Up until recently I have been working with the RAVDESS data set and discovered a new one with over 7000 samples called CREMA. These audio data sets are intended to explore speech emotion recognition. The main tools to explore these datasets have been librosa, Pandas and various classifiers. There is a large number of features that could be extracted from an audio file along with various statistics applied to each feature. This section will explore narrowing down those features and statistics in order to create a set of important features, useful in extracting emotions from audio data. There's also another one called TESS but it has 2800 files so I may dive into that one as well. Below is an exploration mostly of RAVEDESS to get an understanding of the features and how they interact with each other and certain emotions. There is a lot of subtlety between emotions that comes out with various features and I explore some options with RandomForest and GradientBoosting. Currently I have 69 features extracted but hope to narrow it down to about 20.

In these first parts of exploration, I looked at the correlation of features with the '.corr()' method and compared it to the distribution of values plot I had previously. This didn't seem to help much in comparing the differences to emotions so I wanted to see a visual representation and pairplots with Seaborn was my second approach. Because of the large number of features and limitations to Githubs uploads (the Seaborn pairplots added a lot to the size), I had to limit the number of features and comparisons.

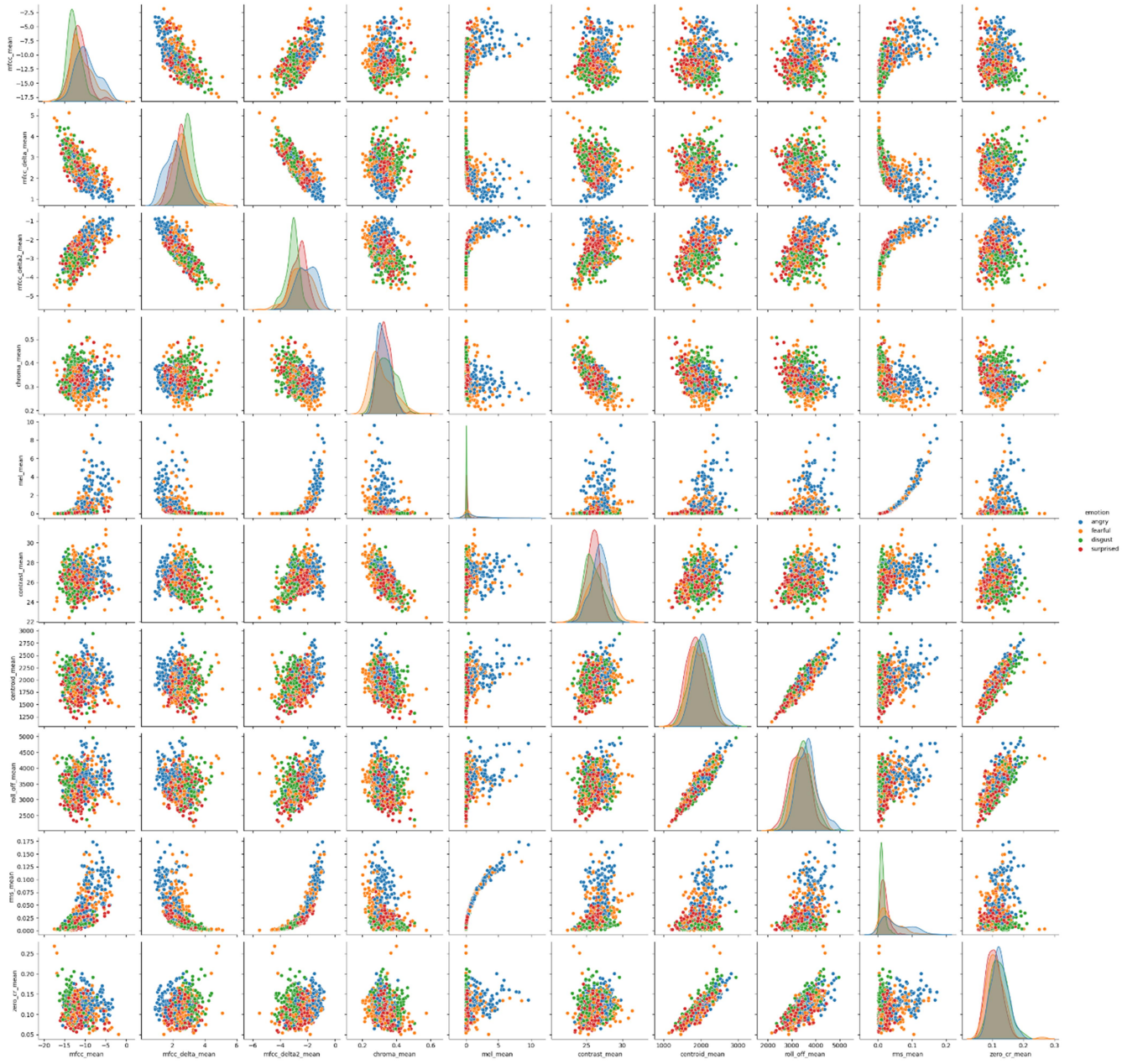
# Mean Neutral Calm Happy Sad



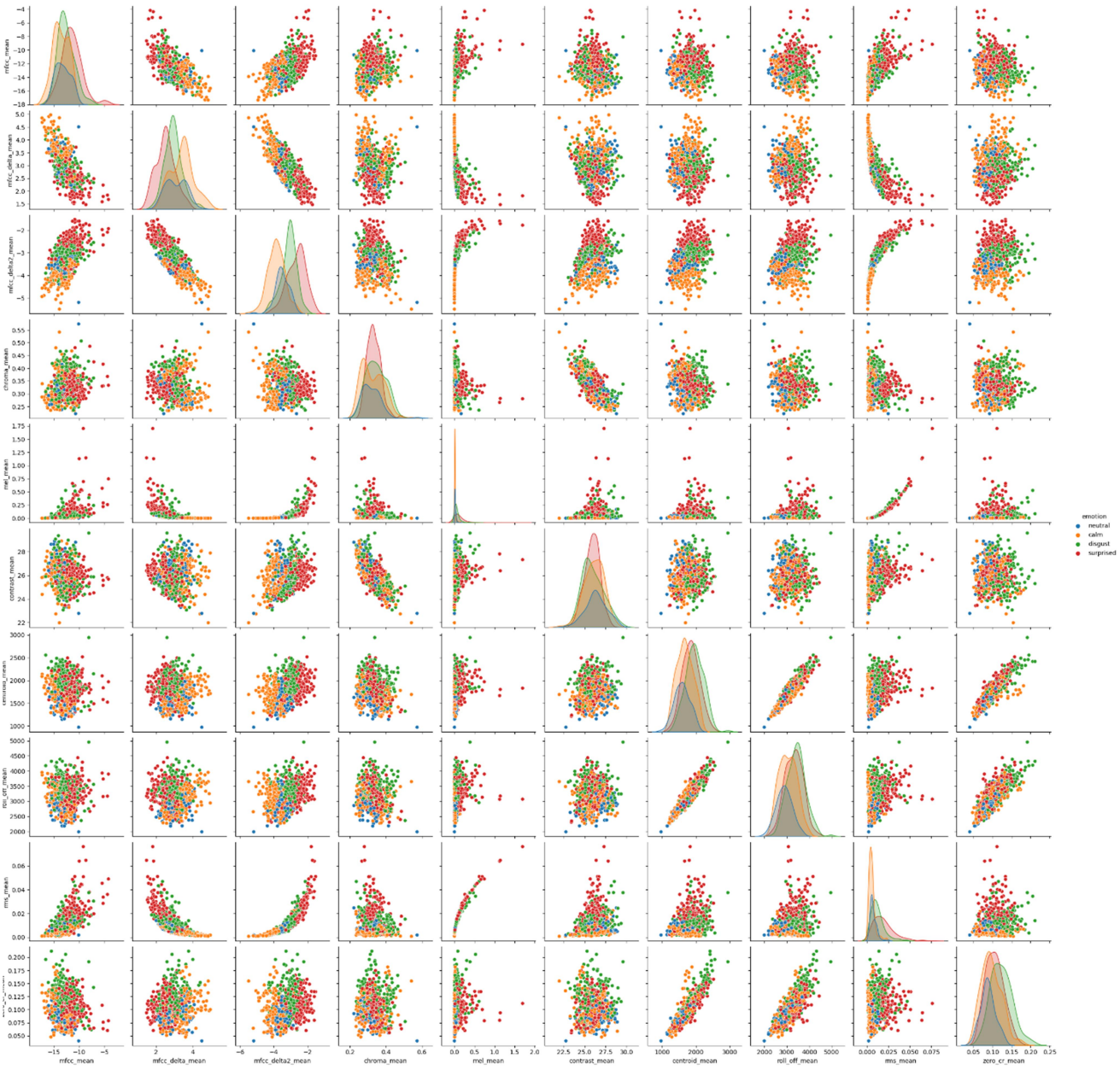
# Mean Happy Sad Angry Fearful



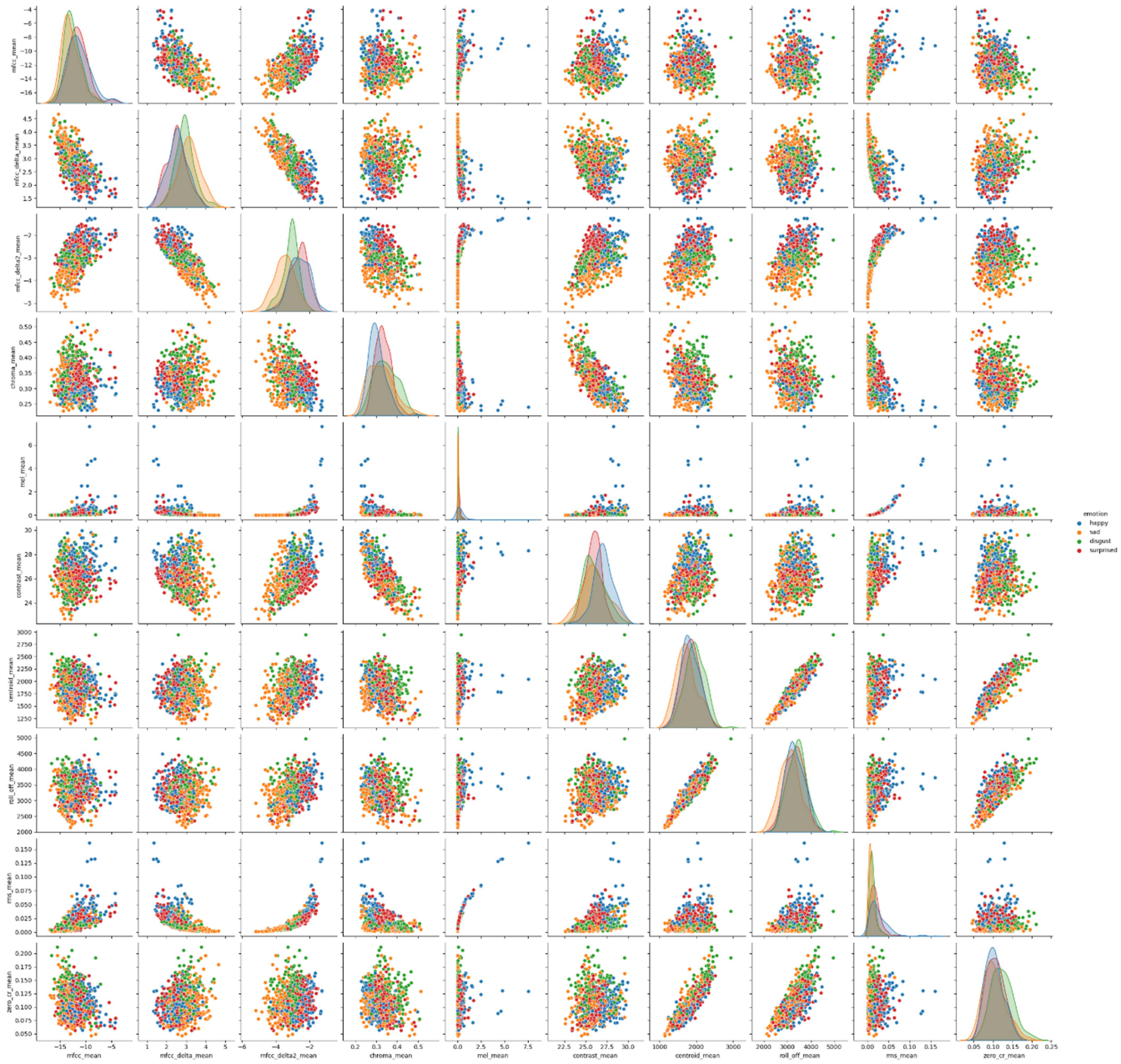
## Mean Angry Fearful Disgust Surprised



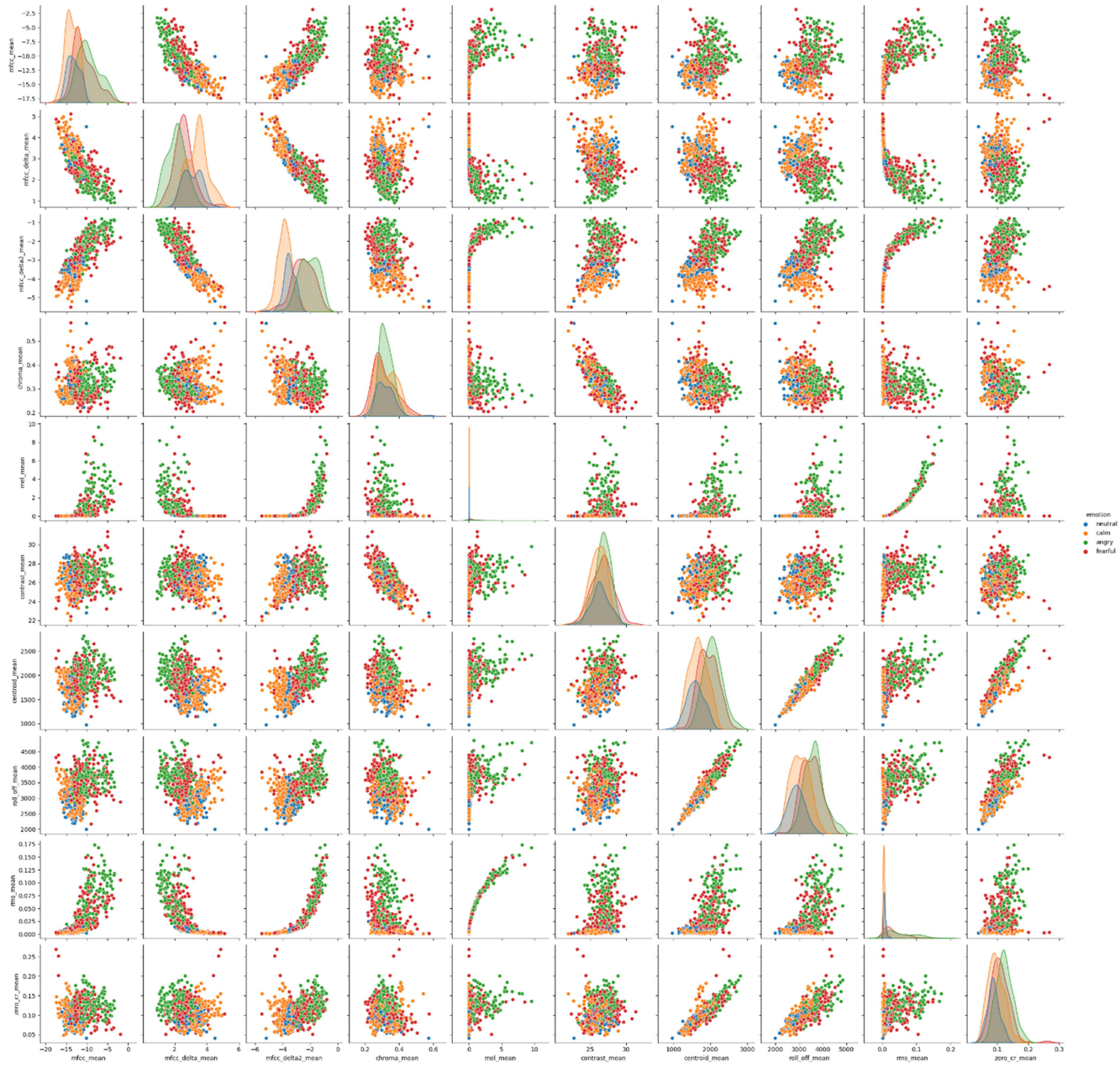
# Mean Disgust Surprised Neutral Calm



# Mean Happy Sad Disgust Surprised



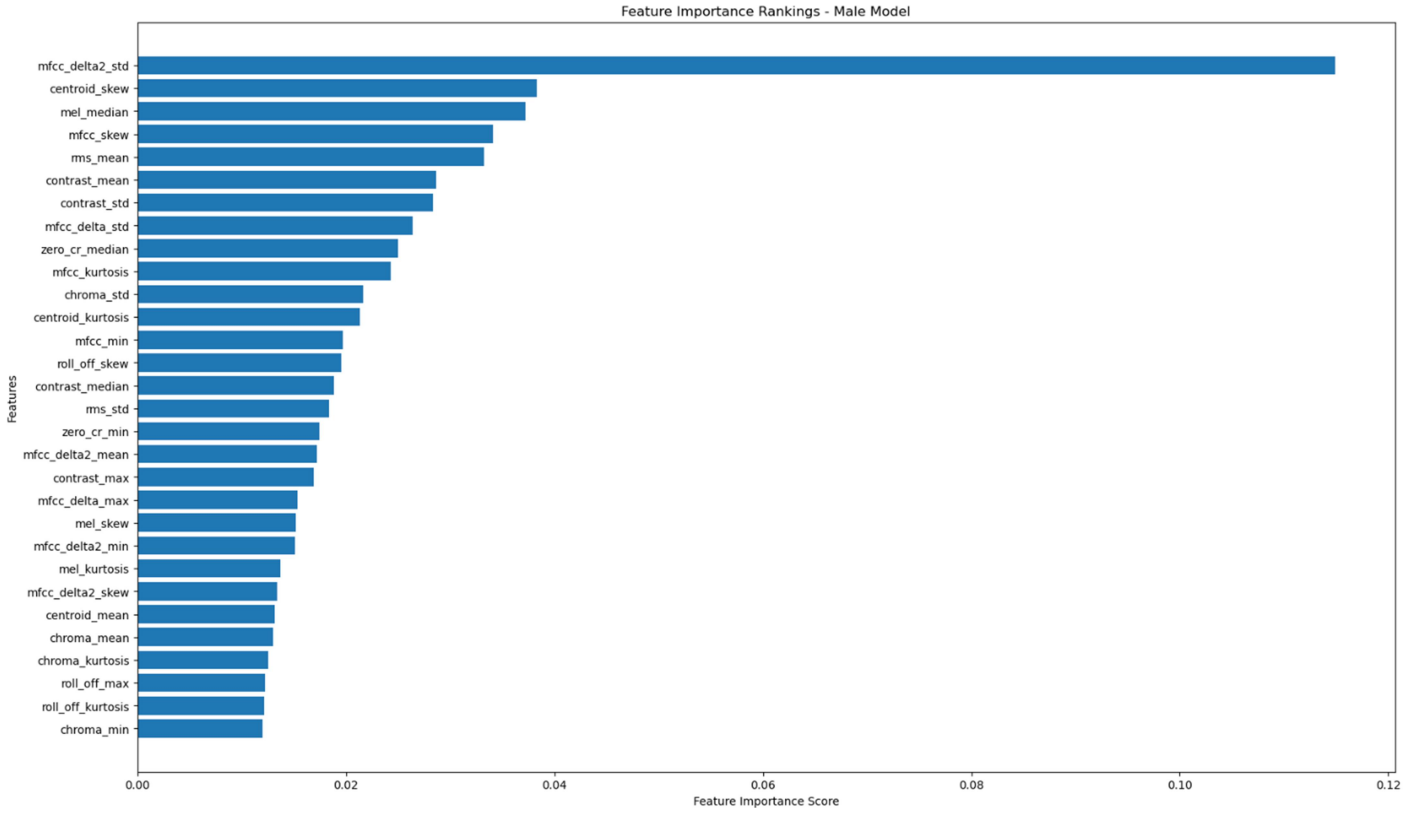
# Mean Neutral Calm Angry Fearful



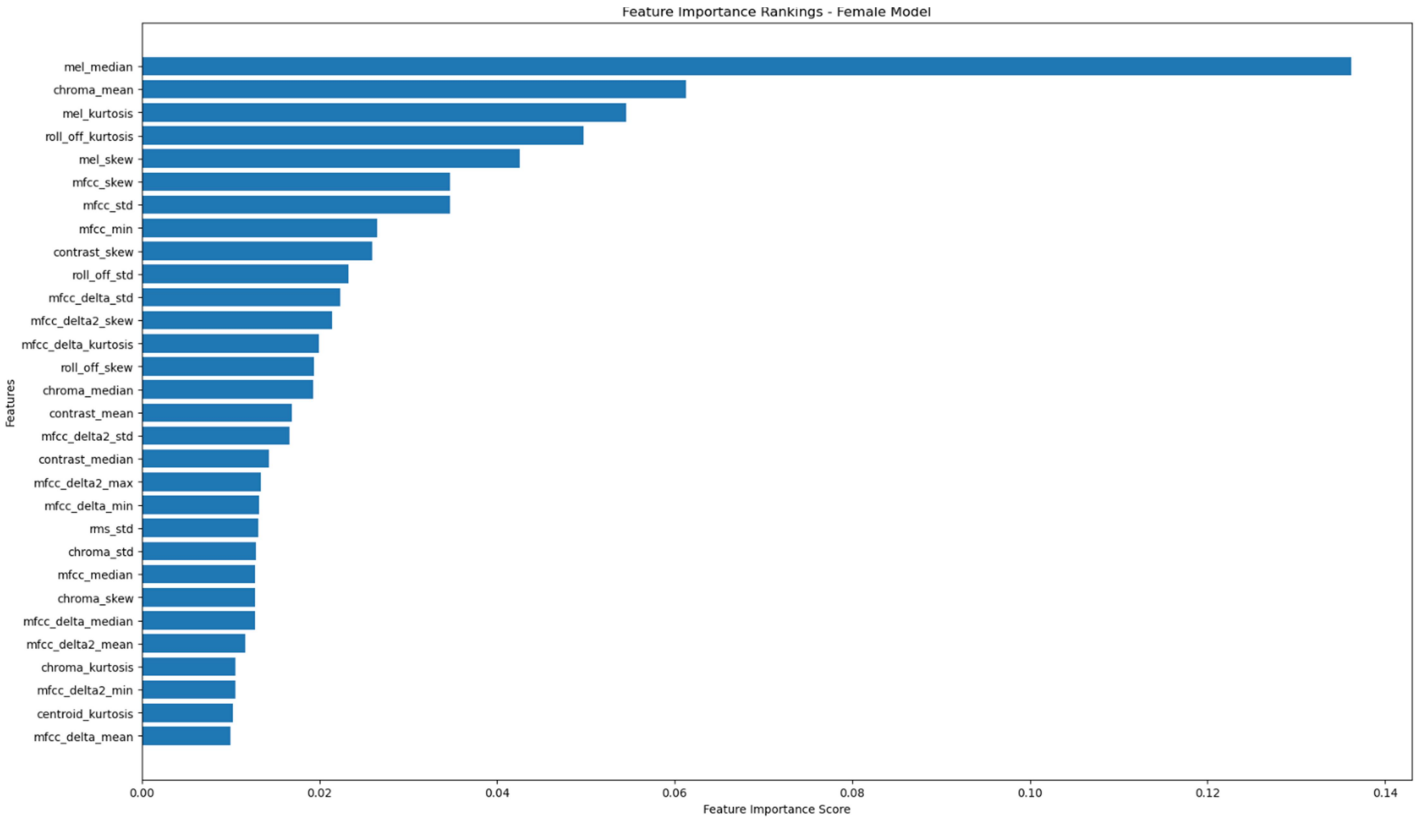
# Exploring features with models

As you can see from the Seaborn pairplots there are some distinct differences between emotions when looking at the mean but there is also a lot of overlap. Instead of trying to compare all 70 features through memory intensive images, I decided to use a model to extract a list of feature impotance. The following is the result of Random Forrest Classifier, Gradient Boosting and feature importance. I also split the sets into male and female voice actors to help determine if there are significant differences in feature importance. I ran separate models for scaled vs. unscaled as well. Here are some of the results:

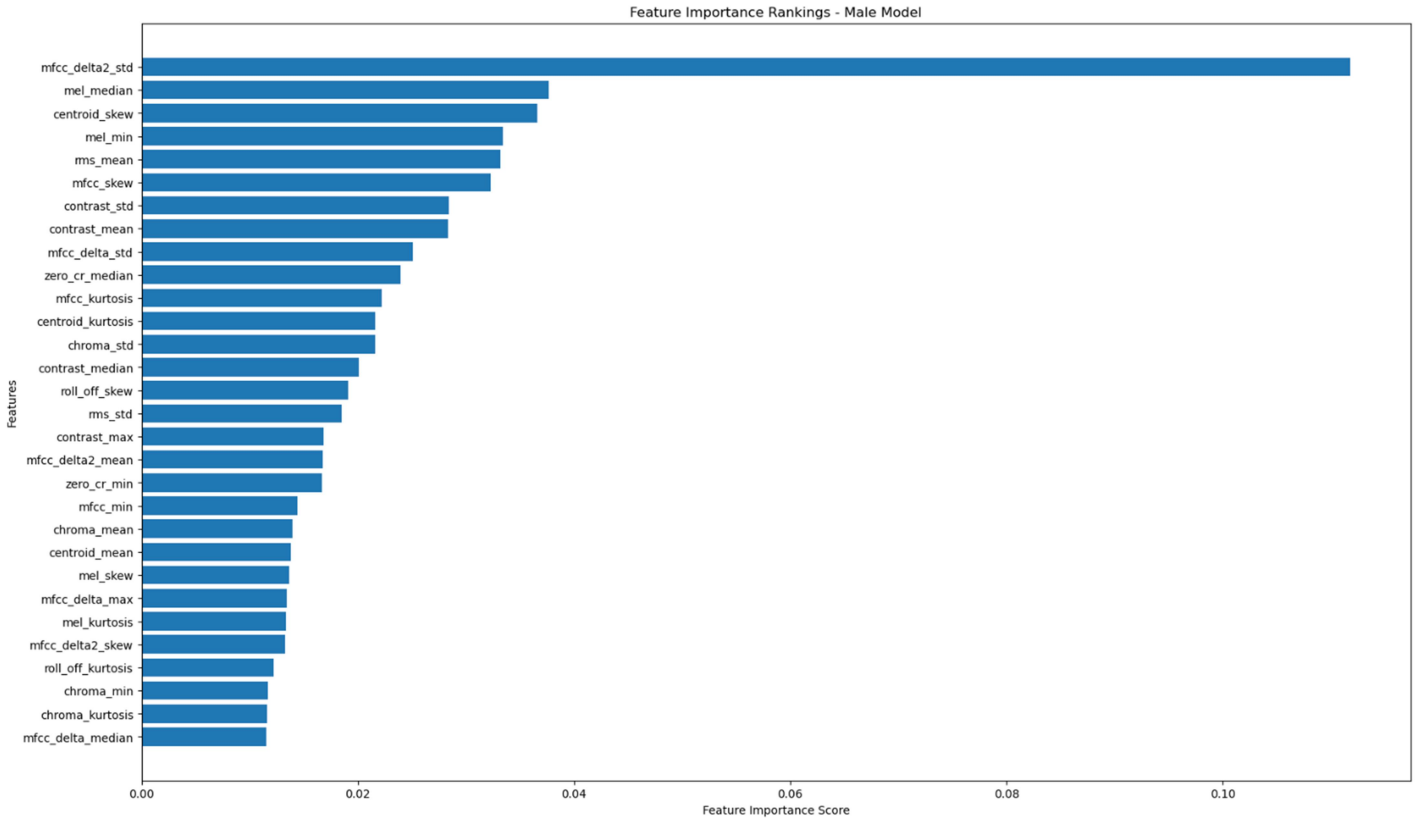
**Feature Importance Rankings - Male Random Forest Model**



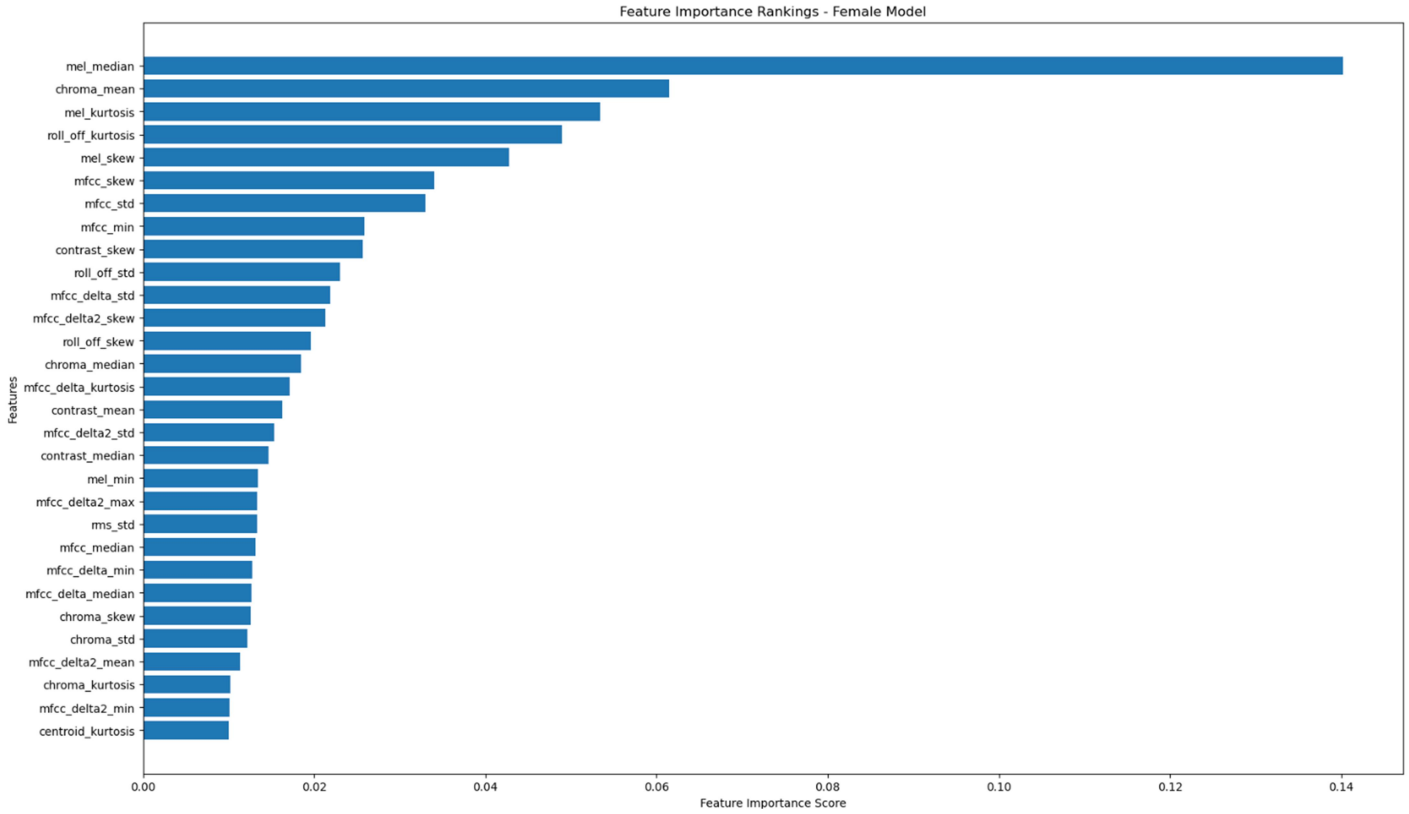
## Feature Importance Rankings - Female Random Forest Model




## Feature Importance Rankings - Male Random Forest Model Scaled

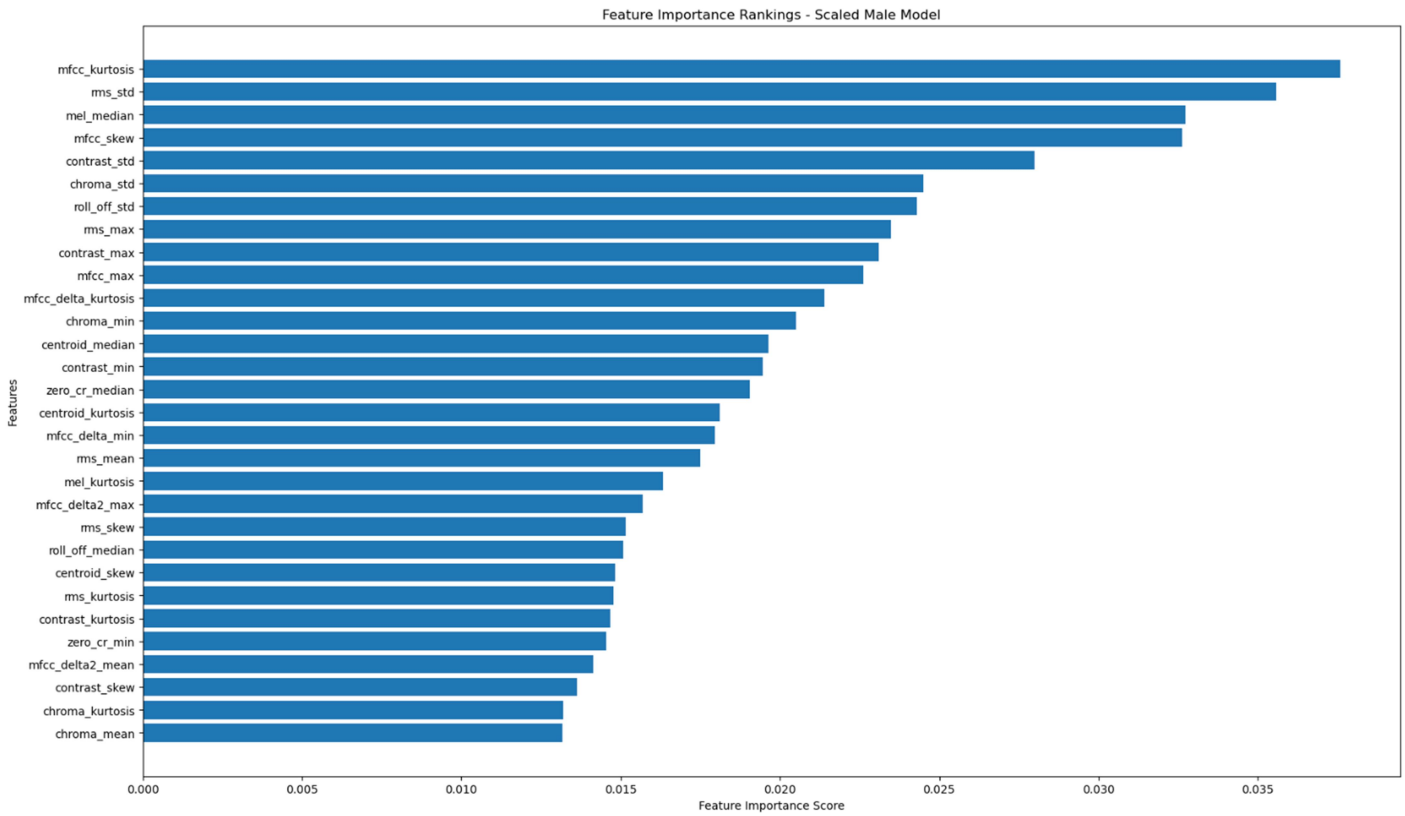


**Feature Importance Rankings - Female Random Forest Model Scaled**

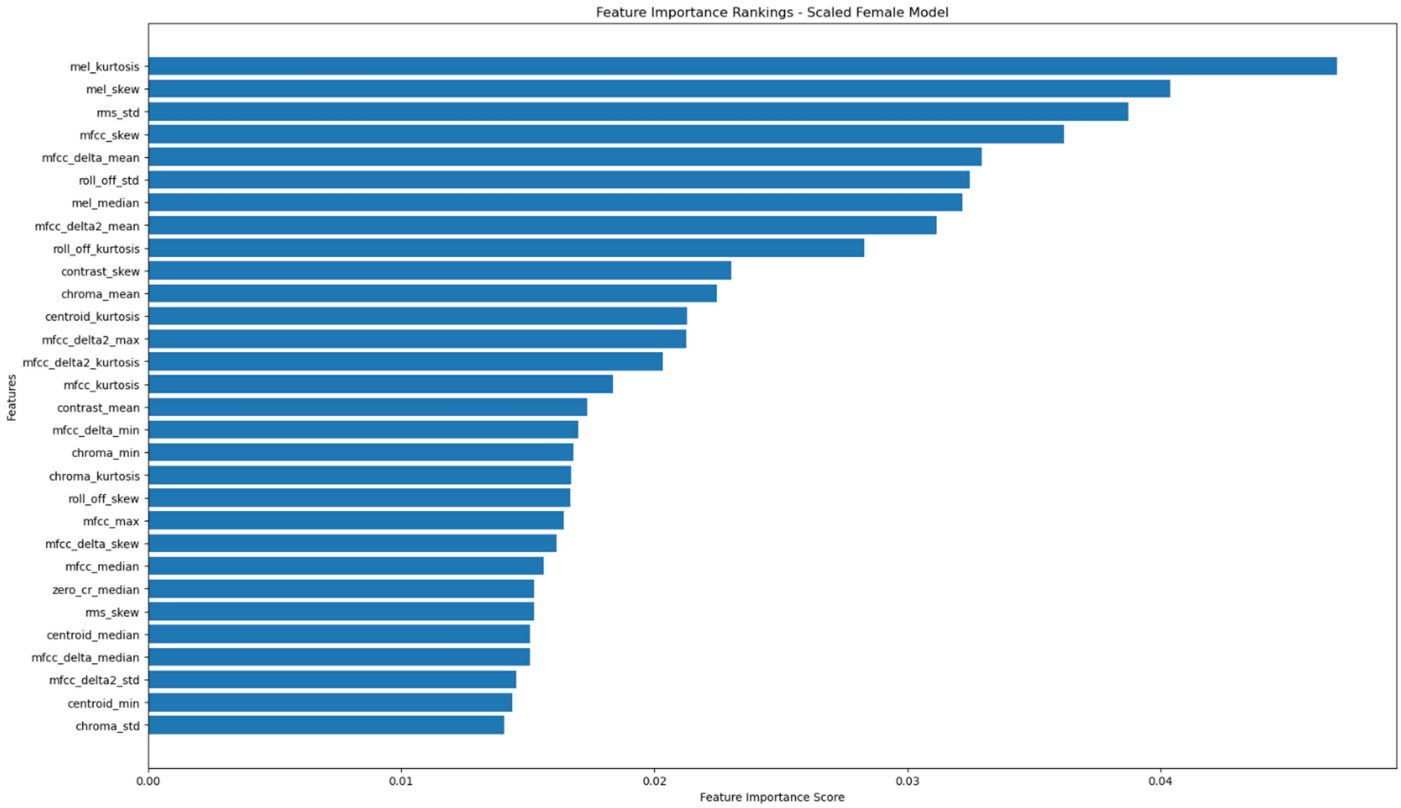



Here are the results of Gradient Boosting:

**Feature Importance Rankings - Male Gradient Boosting Model Scaled**

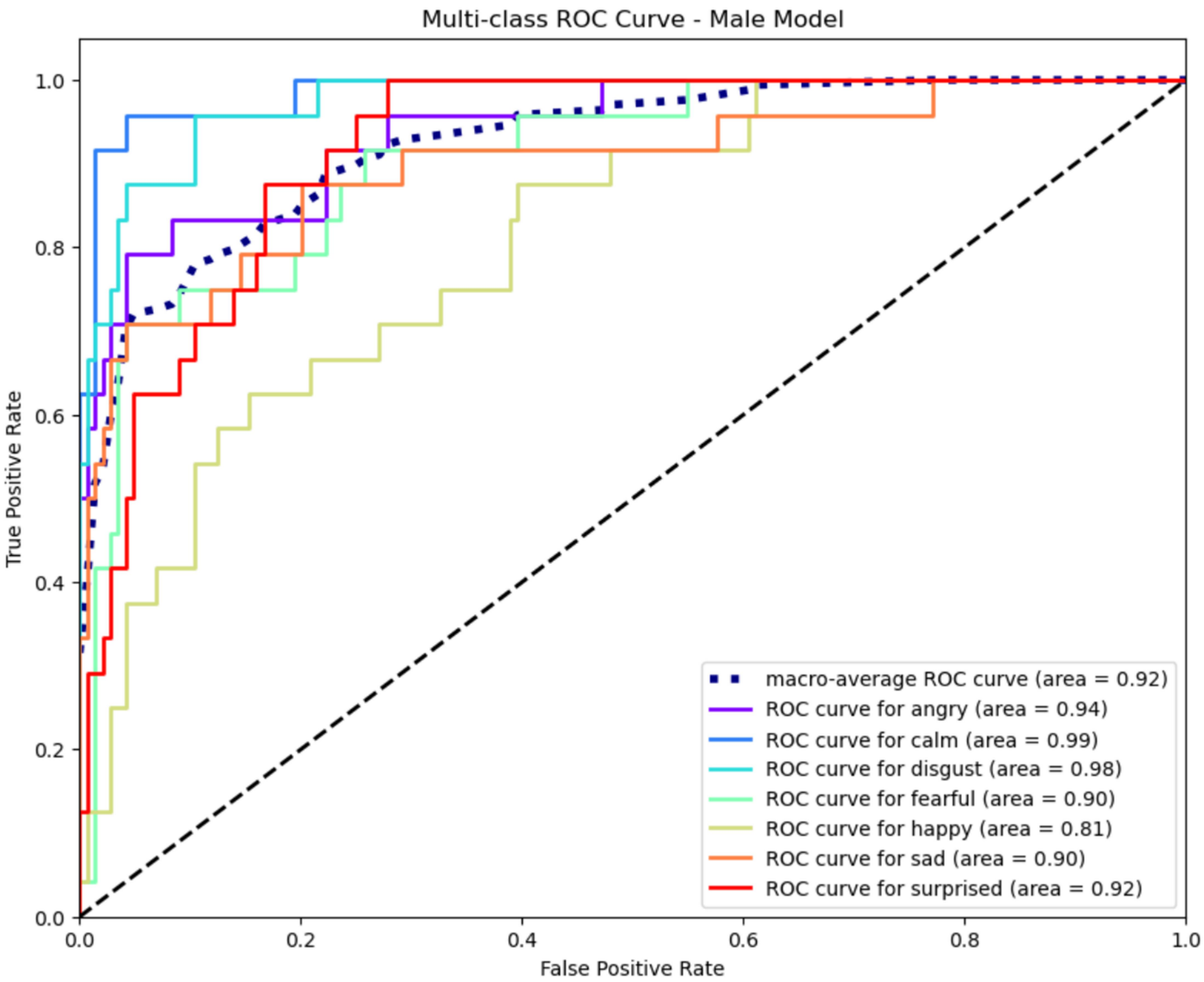


**Feature Importance Rankings - Female Gradient Boosting Model Scaled**

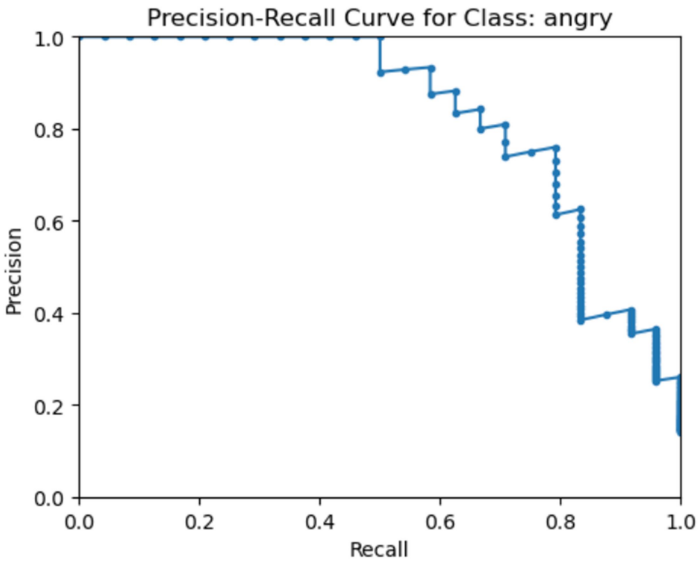


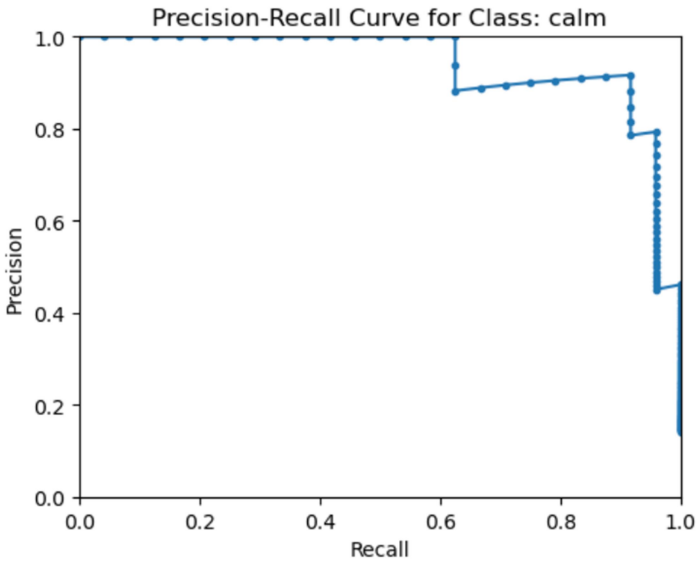

Even though this wasn't the final model I was planning to use, I ran some additional functions to see the ROC Curve and Precision-Recall Curves:

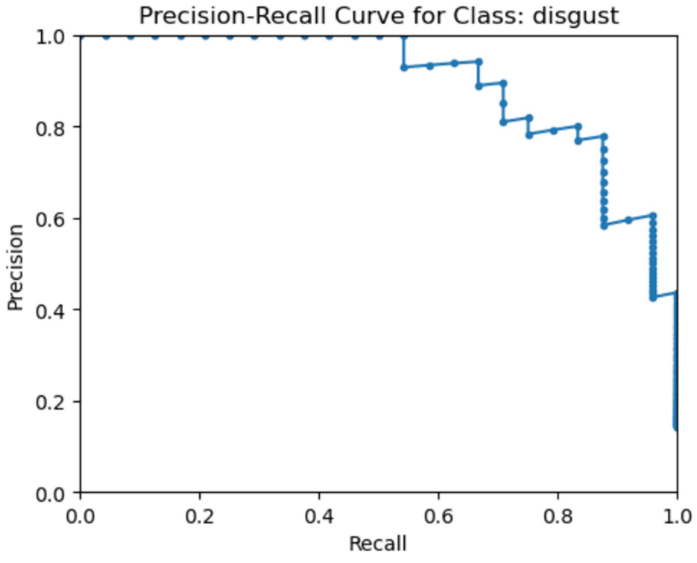
**ROC Curve – Gradient Boosting Male Scaled**

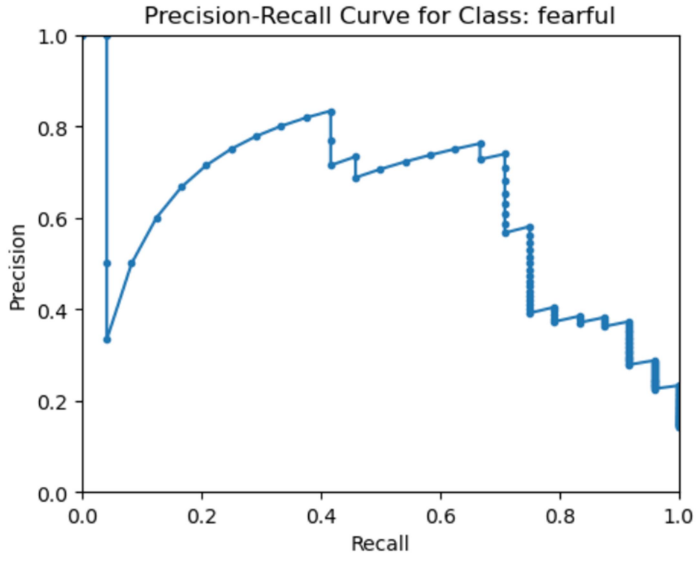


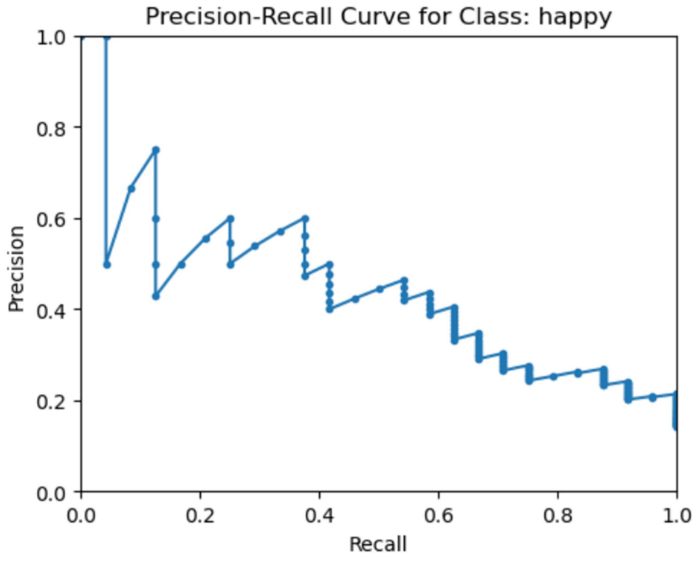
**Precision-Recall Curves – Gradient Boosting Male Scaled**

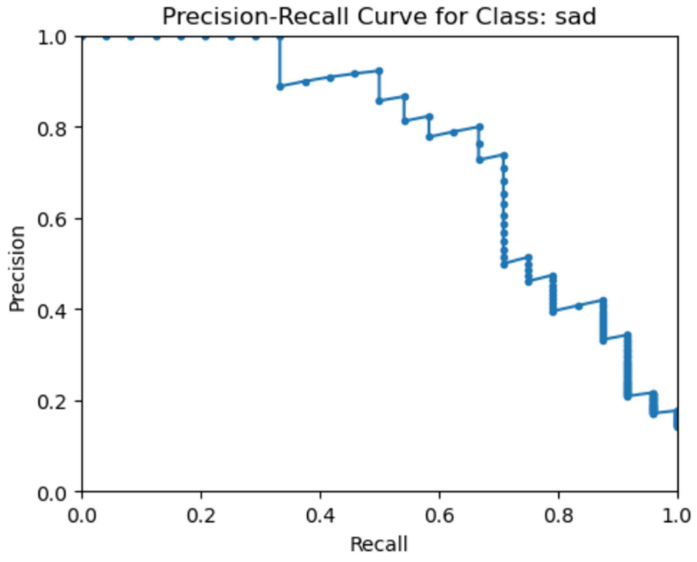


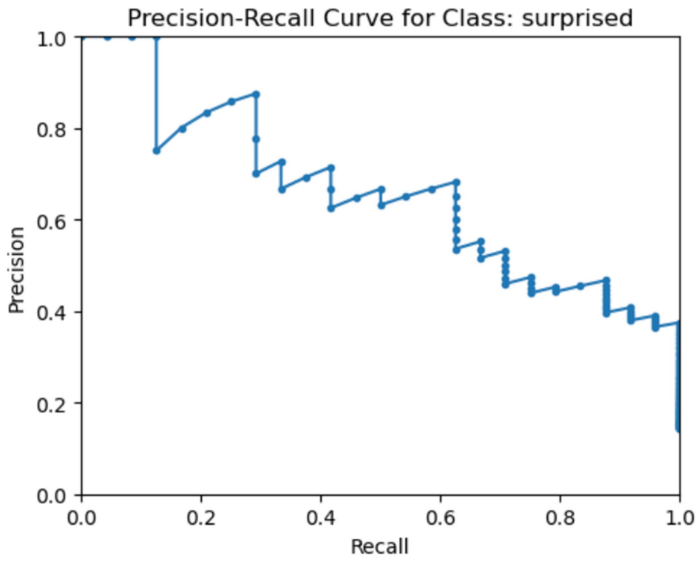






## Feature Importance

Through some initial testing and the pairplots results, there are a few features and statistics that stand out. In general the RandomForest and GradientBoosting reveiled that median, std, mean, skew and kurtosis were the most helpful statistics. A few others like min and max played a role as well but appeared less requently. After several tests, GradientBoosting gave better accuracy scores on average than RandomForest. Some of the best features GradientBoosting displayed were mfcc_delta_median, rms_std, mfcc_delta2_max, contrast_median, and chroma_std for male actors. For female actors the top 5 features from GradientBoosting were mel_kurtosis, mfcc_skew, rms_std, mel_skew, and roll_off_kurtosis.

## EDA Thoughts

While GradientBoosting gave impressive results in the male emotion identification of about 71.42% accuracy, I'm hoping to get better results later from neural net classifiers. The features that were discovered in these test will undoubtedly help in further exploration of this project. The small variation on the pairplots were evitdent but mostly in the stronger emotions such as 'anger' and 'disgust'. Another succesfully detected emotion was 'calm' which makes sense since it will stand out the most from the other emotions as a unique feature shape due to it's lack of change in audio features.

**Some Changes**

After various testing, I decided to move forward with only the RAVDESS dataset. TESS had great performance but only consisted of 2 voice actresses and did not cover the masculine nature of speech. CREMA-D was larger than RAVDESS but due to it's varied nature of nationalities and accents, it seemed best to start with the middle ground. There may be some evidence below of these 2 datasets but they will not be used for final testing.

## Exploring Features with Models

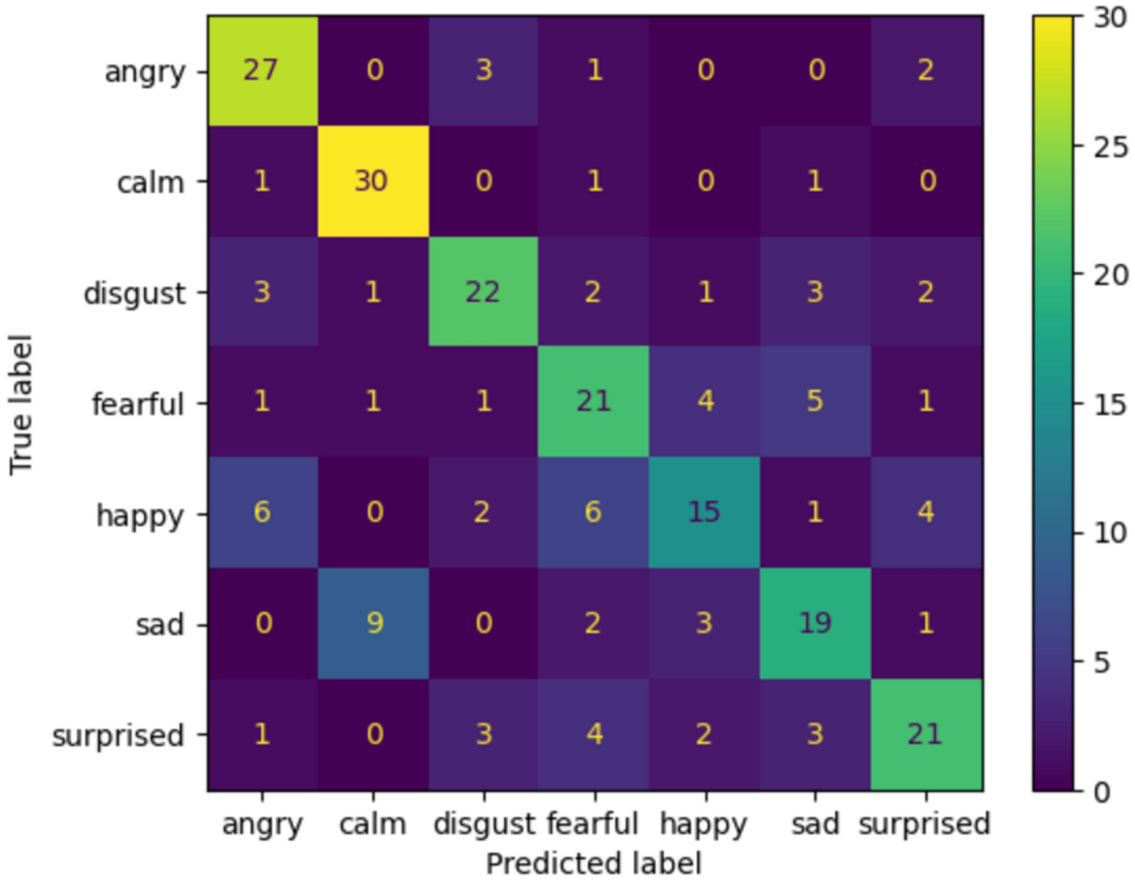
I ran some further testing with Random Forest and Gradient Boosting to target accuracy rather than feature importance. In the process I explored some hyper parameter tuning in preparation for the final model. Here's a confusion matrix and precision-recall scores from the Gradient Boosting:

Confusion Matrix for Male Model:


| | precision | recall | f1-score | support |
|---|---|---|---|---|
| angry | 0.67 | 0.73 | 0.70 | 33 |
| calm | 0.74 | 0.88 | 0.81 | 33 |
| disgust | 0.71 | 0.74 | 0.72 | 34 |
| fearful | 0.67 | 0.59 | 0.62 | 34 |
| happy | 0.52 | 0.47 | 0.49 | 34 |
| sad | 0.58 | 0.53 | 0.55 | 34 |
| surprised | 0.62 | 0.62 | 0.62 | 34 |
| accuracy | | | 0.65 | 236 |
| macro avg | 0.64 | 0.65 | 0.64 | 236 |
| weighted avg | 0.64 | 0.65 | 0.64 | 236 |

Training accuracy 1.0000
Testing accuracy 0.6483

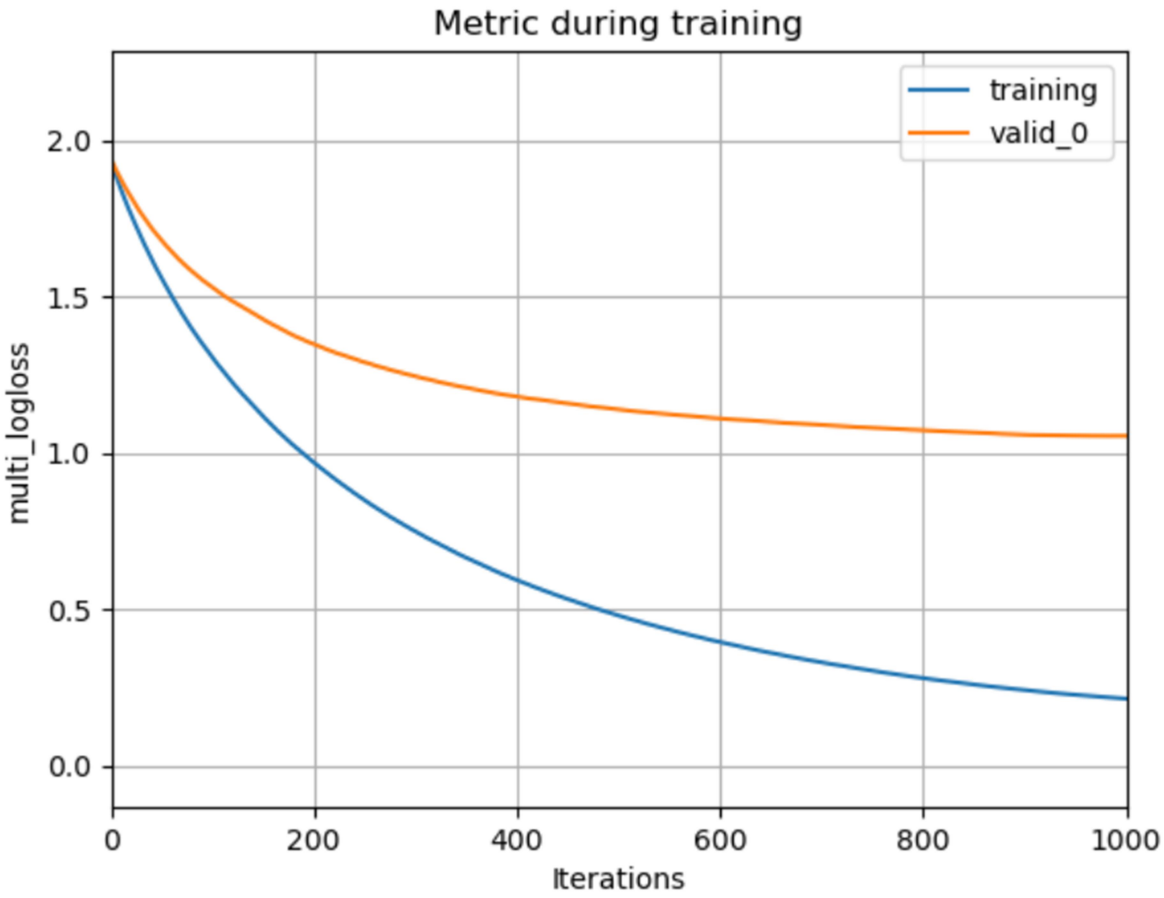
Confusion Matrix for Female Model:

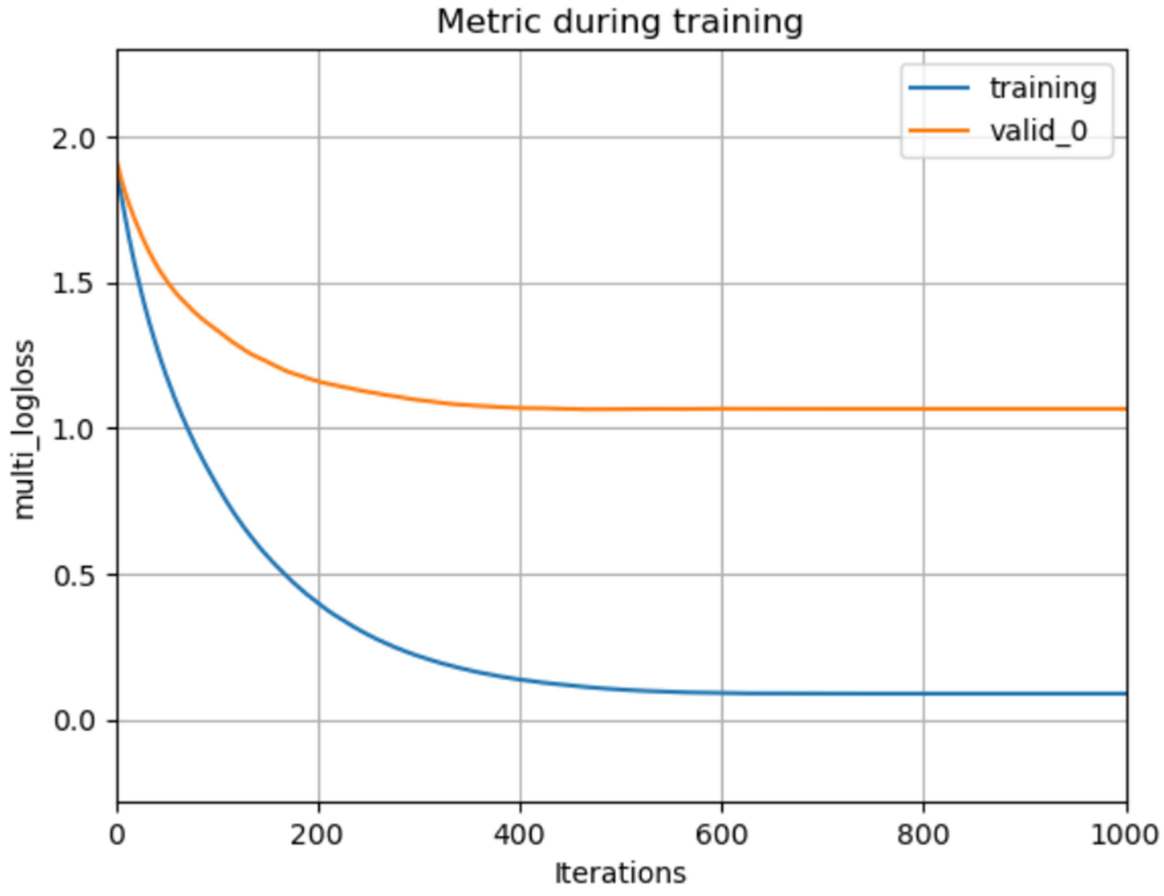



| | precision | recall | f1-score | support |
|---|---|---|---|---|
| angry | 0.69 | 0.82 | 0.75 | 33 |
| calm | 0.73 | 0.91 | 0.81 | 33 |
| disgust | 0.71 | 0.65 | 0.68 | 34 |
| fearful | 0.57 | 0.62 | 0.59 | 34 |
| happy | 0.60 | 0.44 | 0.51 | 34 |
| sad | 0.59 | 0.56 | 0.58 | 34 |
| surprised | 0.68 | 0.62 | 0.65 | 34 |
| accuracy | | | 0.66 | 236 |
| macro avg | 0.65 | 0.66 | 0.65 | 236 |
| weighted avg | 0.65 | 0.66 | 0.65 | 236 |

Training accuracy 1.0000
Testing accuracy 0.6568

One issue I'm sure most encounter during hyper parameter tuning is the time it takes in between each test and tweak. I wanted to capitalize on the multi-core support of Light Gradient Boosting Machine so I gave that a shot as well in hopes to speed up the process. Here are some results:


Metric during training
training
valid_0
multi_logloss
Iterations
0
200
400
600
800
1000
0.0
0.5
1.0
1.5
2.0

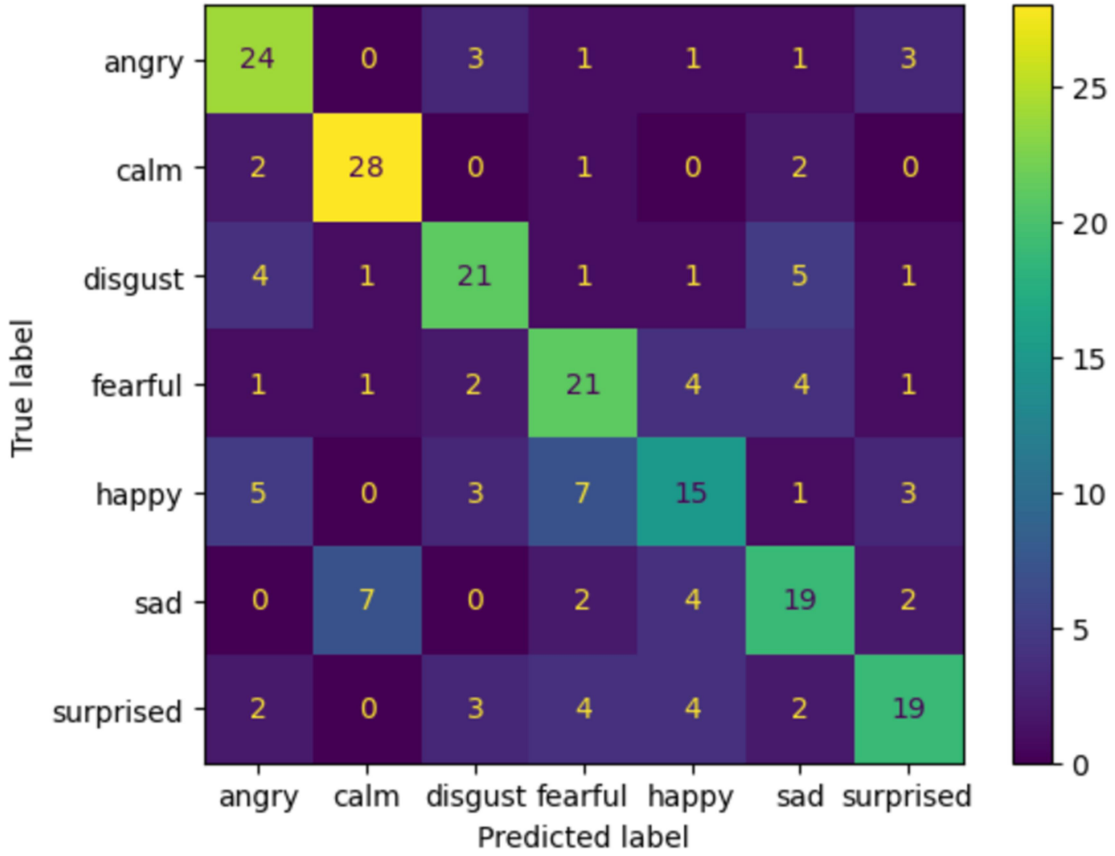

Metric during training
training
valid_0
multi_logloss
Iterations
0
200
400
600
800
1000
0.0
0.5
1.0
1.5
2.0

Confusion Matrix for Male Model:


| | precision | recall | f1-score | support |
|---|---|---|---|---|
| angry | 0.66 | 0.70 | 0.68 | 33 |
| calm | 0.67 | 0.79 | 0.72 | 33 |
| disgust | 0.69 | 0.71 | 0.70 | 34 |
| fearful | 0.53 | 0.53 | 0.53 | 34 |
| happy | 0.59 | 0.50 | 0.54 | 34 |
| sad | 0.57 | 0.47 | 0.52 | 34 |
| surprised | 0.56 | 0.59 | 0.57 | 34 |
| accuracy | | | 0.61 | 236 |
| macro avg | 0.61 | 0.61 | 0.61 | 236 |
| weighted avg | 0.61 | 0.61 | 0.61 | 236 |

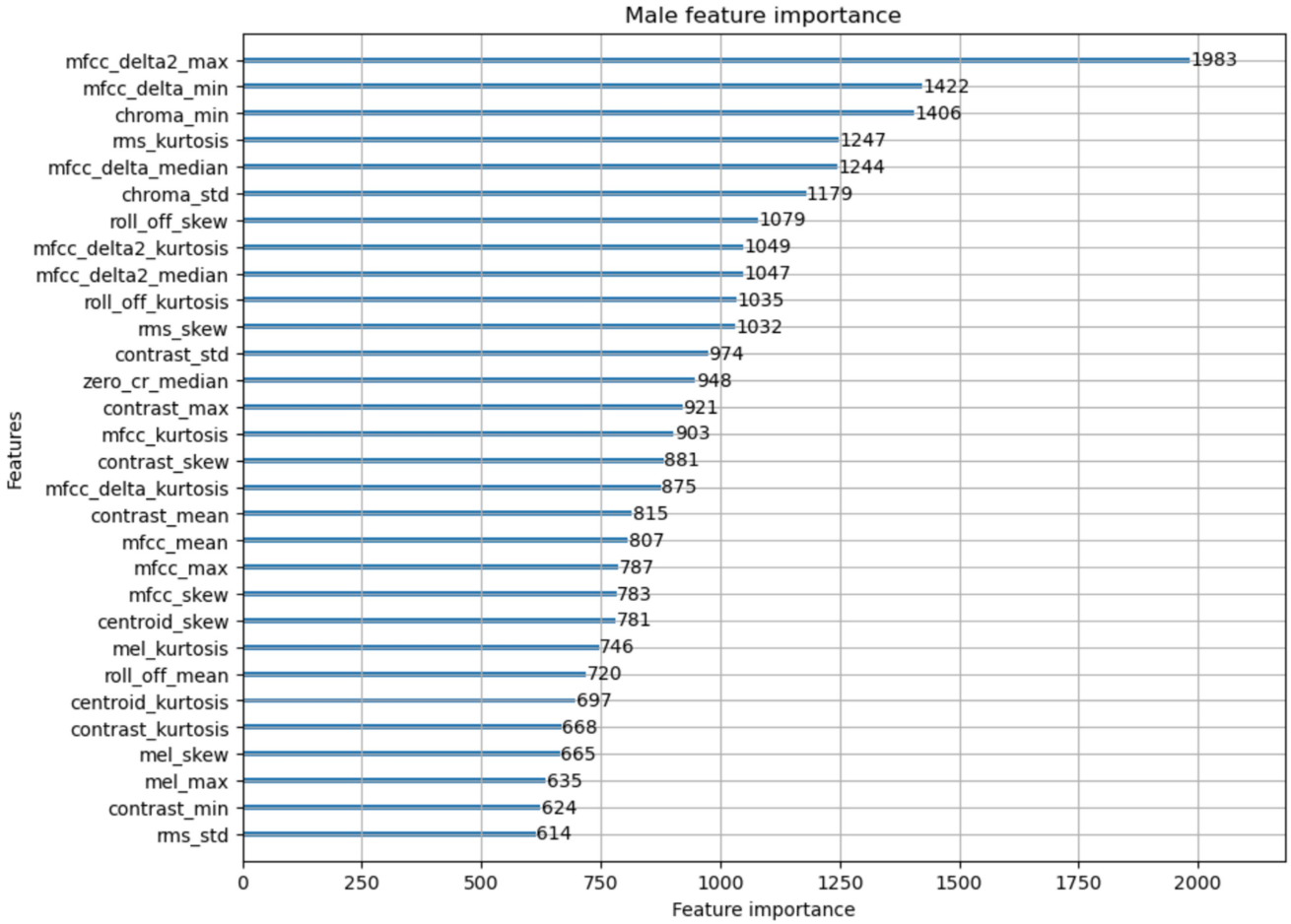
Training accuracy 1.0000
Testing accuracy 0.6102

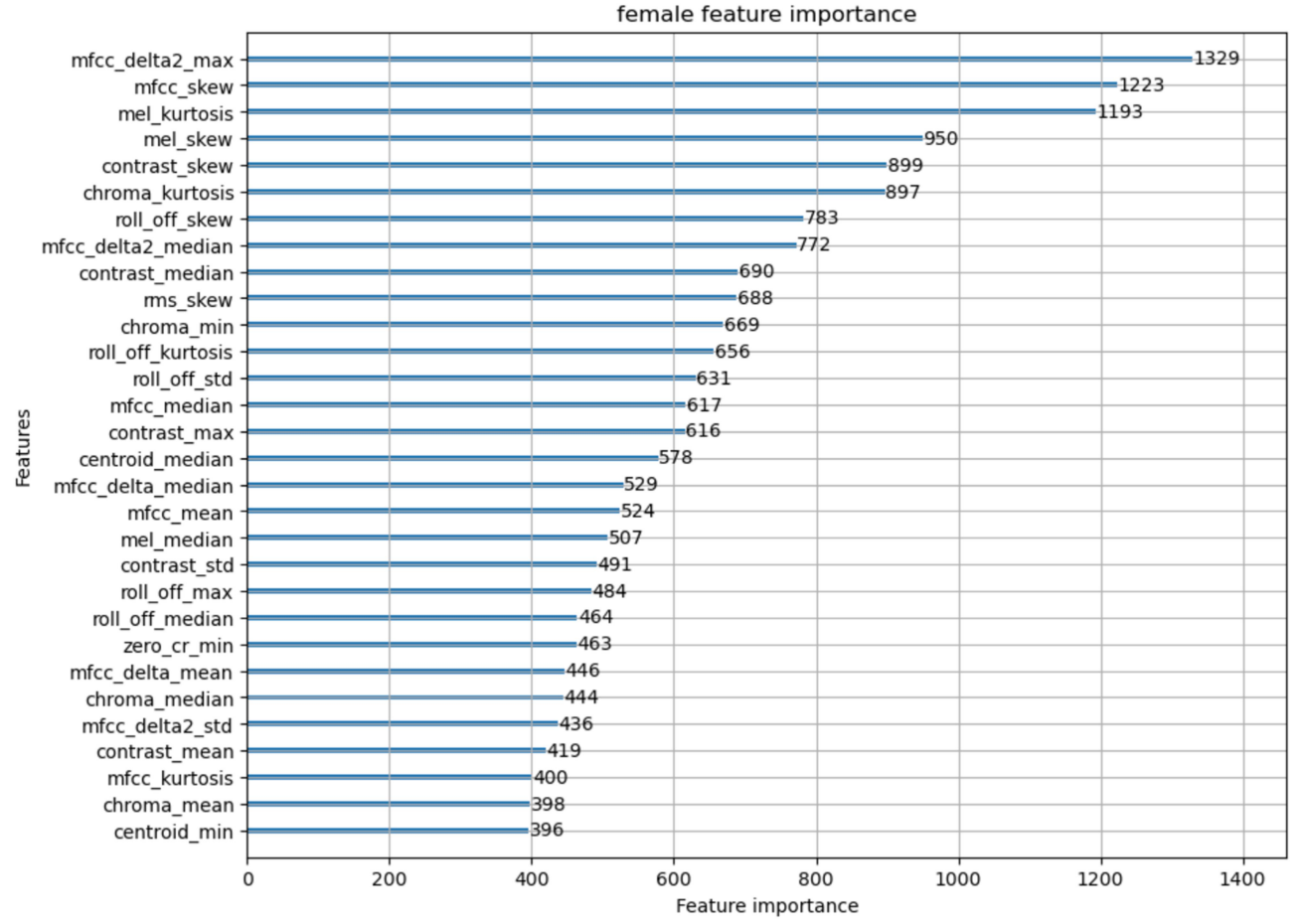
Confusion Matrix for Female Model:



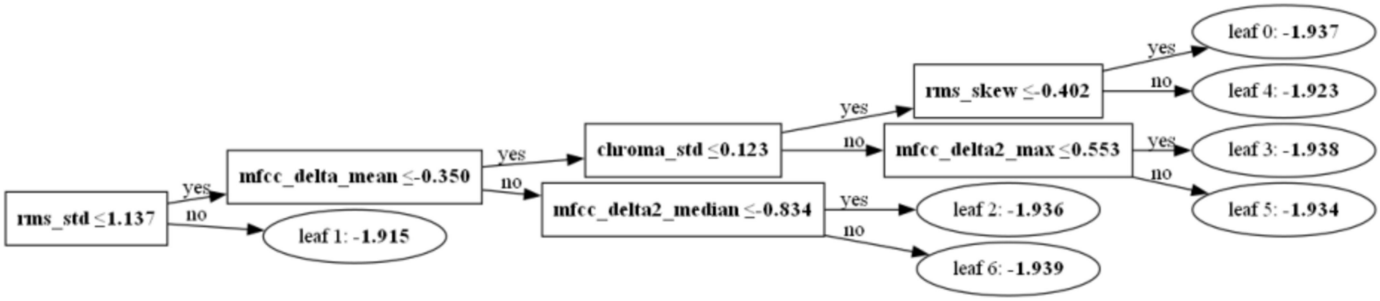

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| angry | 0.63 | 0.73 | 0.68 | 33 |
| calm | 0.76 | 0.85 | 0.80 | 33 |
| disgust | 0.66 | 0.62 | 0.64 | 34 |
| fearful | 0.57 | 0.62 | 0.59 | 34 |
| happy | 0.52 | 0.44 | 0.48 | 34 |
| sad | 0.56 | 0.56 | 0.56 | 34 |
| surprised | 0.66 | 0.56 | 0.60 | 34 |
| accuracy | | | 0.62 | 236 |
| macro avg | 0.62 | 0.62 | 0.62 | 236 |
| weighted avg | 0.62 | 0.62 | 0.62 | 236 |

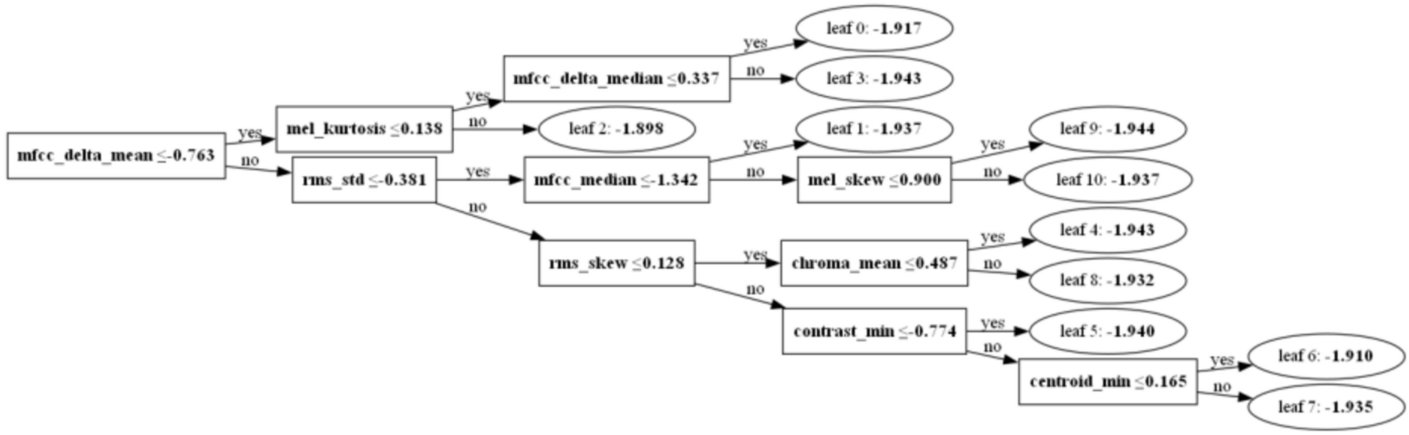
Training accuracy 1.0000
Testing accuracy 0.6229


Male feature importance
mfcc_delta2_max 1983
mfcc_delta_min 1422
chroma_min 1406
rms_kurtosis 1247
mfcc_delta_median 1244
chroma_std 1179
roll_off_skew 1079
mfcc_delta2_kurtosis 1049
mfcc_delta2_median 1047
roll_off_kurtosis 1035
rms_skew 1032
contrast_std 974
zero_cr_median 948
contrast_max 921
mfcc_kurtosis 903
contrast_skew 881
mfcc_delta_kurtosis 875
contrast_mean 815
mfcc_mean 807
mfcc_max 787
mfcc_skew 783
centroid_skew 781
mel_kurtosis 746
roll_off_mean 720
centroid_kurtosis 697
contrast_kurtosis 668
mel_skew 665
mel_max 635
contrast_min 624
rms_std 614
Features
0 250 500 750 1000 1250 1500 1750 2000
Feature importance


female feature importance
mfcc_delta2_max 1329
mfcc_skew 1223
mel_kurtosis 1193
mel_skew 950
contrast_skew 899
chroma_kurtosis 897
roll_off_skew 783
mfcc_delta2_median 772
contrast_median 690
rms_skew 688
chroma_min 669
roll_off_kurtosis 656
roll_off_std 631
mfcc_median 617
contrast_max 616
centroid_median 578
mfcc_delta_median 529
mfcc_mean 524
mel_median 507
contrast_std 491
roll_off_max 484
roll_off_median 464
zero_cr_min 463
mfcc_delta_mean 446
chroma_median 444
mfcc_delta2_std 436
contrast_mean 419
mfcc_kurtosis 400
chroma_mean 398
centroid_min 396
Features
0 200 400 600 800 1000 1200 1400
Feature importance

Here are some of the decision tree paths that LGBM took when modeling which I though were interesting.




Since Gradient Boosting performed well I wanted to try some further testing with Light Gradient Boosting but did not see a significant improvement even when mixing in some RandomizedSearchCV hyper-parameter variations. mfcc_delta2_max, mfcc_delta_min, and chroma_min were high performers for male voices. mfcc_delta2_max, mfcc_skew, and mel_kurtosis were high performers for female voices.

## Thoughts on these models

These testing models helped with identifying some good features so we can narrow down the 70+ that were extracted in the dataframe creation process. I will use these features going forward to help tune a different model in hopes of getting higher performance. Male voice features that performed well for these 3 models were: mel_median, mfcc_delta_median, mfcc_delta2_max, mfcc_delta2_min, mfcc_delta_min, chroma_min, mfcc_std, rms_std, chroma_std, and rms_std. For female voices, the features the performed well with these models were more consistent: mel_median, mel_skew, mfcc_delta2_max, roll_off_kertosis, mfcc_skew, and mel_kurtosis. I'm interested to see if the important features for the female voices will have better resutls.

# Modeling

During this portion I will be exploring different hyper-parameters and the final model to evaluate the emotion classes. A few thoughts, the datasets available are not very abundant and emotion interpretation seems like an arbitrary subject. It would be interesting to see how results differed if we had a large dataset with professional quality labeled emotion samples with varying languages, dialects, races, sexes and accents. Maybe in the future someone can tackle that since I could not find it. Even with such a dataset, I wonder if human experts would even be able to interpret the emotions correctly.

# Extracting Features into Train Test sets Directly

This was the old method that I found in the article on data-flair.training. I expanded on it to include additional features from Librosa and more emotional classes to give a wider range to detect. Here are the results of this Multi-Layer Perceptron Classifier with this extraction method:

Example Classification Report for male scaled:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| angry | 0.91 | 0.83 | 0.87 | 24 |
| calm | 0.79 | 1.00 | 0.88 | 26 |
| disgust | 0.56 | 0.88 | 0.68 | 16 |
| fearful | 0.59 | 0.52 | 0.55 | 25 |
| happy | 0.67 | 0.40 | 0.50 | 25 |
| sad | 0.69 | 0.65 | 0.67 | 31 |
| surprised | 0.77 | 0.81 | 0.79 | 21 |
| accuracy | | | 0.71 | 168 |
| macro avg | 0.71 | 0.73 | 0.71 | 168 |
| weighted avg | 0.72 | 0.71 | 0.70 | 168 |

Example Confusion Matrix for male scaled:

```
[[20  0  2  1  0  0  1]
 [ 0 26  0  0  0  0  0]
 [ 0  0 14  0  0  1  1]
 [ 0  1  3 13  2  5  1]
 [ 1  2  1  6 10  3  2]
 [ 1  4  3  2  1 20  0]
 [ 0  0  2  0  2  0 17]]
```

Accuracy of male scaled: 71.43%

Example Classification Report for female scaled:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| angry | 0.88 | 0.75 | 0.81 | 28 |
| calm | 0.89 | 0.83 | 0.86 | 30 |
| disgust | 0.53 | 0.47 | 0.50 | 19 |
| fearful | 0.74 | 0.71 | 0.72 | 24 |
| happy | 0.70 | 0.73 | 0.72 | 26 |
| sad | 0.60 | 0.79 | 0.68 | 19 |
| surprised | 0.67 | 0.73 | 0.70 | 22 |
| accuracy | | | 0.73 | 168 |
| macro avg | 0.72 | 0.72 | 0.71 | 168 |
| weighted avg | 0.73 | 0.73 | 0.73 | 168 |

Example Confusion Matrix for female scaled:

```
[[21  0  2  0  3  0  2]
 [ 0 25  0  0  0  5  0]
 [ 2  1  9  3  2  1  1]
 [ 0  0  1 17  2  1  3]
 [ 1  0  1  1 19  2  2]
 [ 0  2  1  1  0 15  0]
 [ 0  0  3  1  1  1 16]]
```

Accuracy of female scaled: 72.62%

# Thoughts on this Method

This was a simple example I found online to start this project (https://data-flair.training/blogs/python-mini-project-speech-emotion-recognition/). It was to extract features but they were limited to just a few metrics and only an outline of the possibilities of the project. I expanded on the process to, in hopes, get better results. The addition of the new features helped achieve a similar accuracy but with 7 out of the 8 classes instead of just 4. While it's still not achieving great results, it is an improvement in my eyes. One downside that I just learned from my mentor is that this process is flawed due to data leakage. There are repeated phrases and without properly splitting the actors for the train and test sets, the model will identify the actor's previous statements and falsely increase results.

# GridSearchCV

In these next few sections I'm wanted to find the best hyper-parameters with GridSearch and RandomizedSearch for the final model and compare it to the MLPC. Here's the results of GridSearch with male dataset:

```
Best Parameters: {'mlp__activation': 'tanh', 'mlp__alpha': 1.4, 'mlp__batch_size': 163, 'mlp__hidden_layer_sizes': (600,), 'mlp
__learning_rate_init': 0.0003, 'mlp__max_iter': 2000, 'mlp__solver': 'adam'}
Best Score: 0.4603174603174603
[[13  1  3  2  3  2  0]
 [ 0 19  1  0  0  4  0]
 [ 1  4 13  1  0  5  0]
 [ 0  5  0 12  5  1  1]
 [ 0  3  0  1 16  4  0]
 [ 0 11  0  3  2  8  0]
 [ 0  0  4  0  4  7  9]]
              precision    recall  f1-score   support

       angry       0.93      0.54      0.68        24
        calm       0.44      0.79      0.57        24
     disgust       0.62      0.54      0.58        24
     fearful       0.63      0.50      0.56        24
       happy       0.53      0.67      0.59        24
         sad       0.26      0.33      0.29        24
   surprised       0.90      0.38      0.53        24

    accuracy                           0.54       168
   macro avg       0.62      0.54      0.54       168
weighted avg       0.62      0.54      0.54       168
```

## RandomizedSearch with Female Dataset

```
Best Parameters: {'mlp__solver': 'adam', 'mlp__max_iter': 2000, 'mlp__learning_rate_init': 0.0001, 'mlp__hidden_layer_sizes':
(1200,), 'mlp__batch_size': 163, 'mlp__alpha': 0.43333333333333335, 'mlp__activation': 'tanh'}
Best Score: 0.4384920634920635
[[14  1  3  2  1  2  1]
 [ 0 17  0  0  0  7  0]
 [ 1  2 14  1  0  6  0]
 [ 1  1  0 12  5  4  1]
 [ 0  2  0  1 15  6  0]
 [ 0 10  1  1  2 10  0]
 [ 0  0  1  1  4  9  9]]
              precision    recall  f1-score   support

       angry       0.88      0.58      0.70        24
        calm       0.52      0.71      0.60        24
     disgust       0.74      0.58      0.65        24
     fearful       0.67      0.50      0.57        24
       happy       0.56      0.62      0.59        24
         sad       0.23      0.42      0.29        24
   surprised       0.82      0.38      0.51        24

    accuracy                           0.54       168
   macro avg       0.63      0.54      0.56       168
weighted avg       0.63      0.54      0.56       168
```

# RandomizedSearch with Male Dataset

```
Best Parameters: {'mlp__solver': 'lbfgs', 'mlp__max_iter': 2000, 'mlp__learning_rate_init': 0.00025, 'mlp__hidden_layer_sizes':
(1200,), 'mlp__batch_size': 163, 'mlp__alpha': 1.135, 'mlp__activation': 'tanh'}
Best Score: 0.5595238095238095
[[13  0  2  5  3  0  1]
 [ 1 10  0  4  1  6  2]
 [ 4  0  9  0  5  4  2]
 [ 0  0  0 13  9  0  2]
 [ 0  0  0  7 15  0  2]
 [ 2  1  4  6  3  7  1]
 [ 1  0  1  5  5  0 12]]
              precision    recall  f1-score   support

       angry       0.62      0.54      0.58        24
        calm       0.91      0.42      0.57        24
     disgust       0.56      0.38      0.45        24
     fearful       0.33      0.54      0.41        24
       happy       0.37      0.62      0.46        24
         sad       0.41      0.29      0.34        24
   surprised       0.55      0.50      0.52        24

    accuracy                           0.47       168
   macro avg       0.53      0.47      0.48       168
weighted avg       0.53      0.47      0.48       168
```

As you can see the results were not very good. In various other articles I saw that others were having more success with neural net algorithms so I was pretty sure I was headed in that direction.

# Multi-Layer Perceptron Classifier New Method

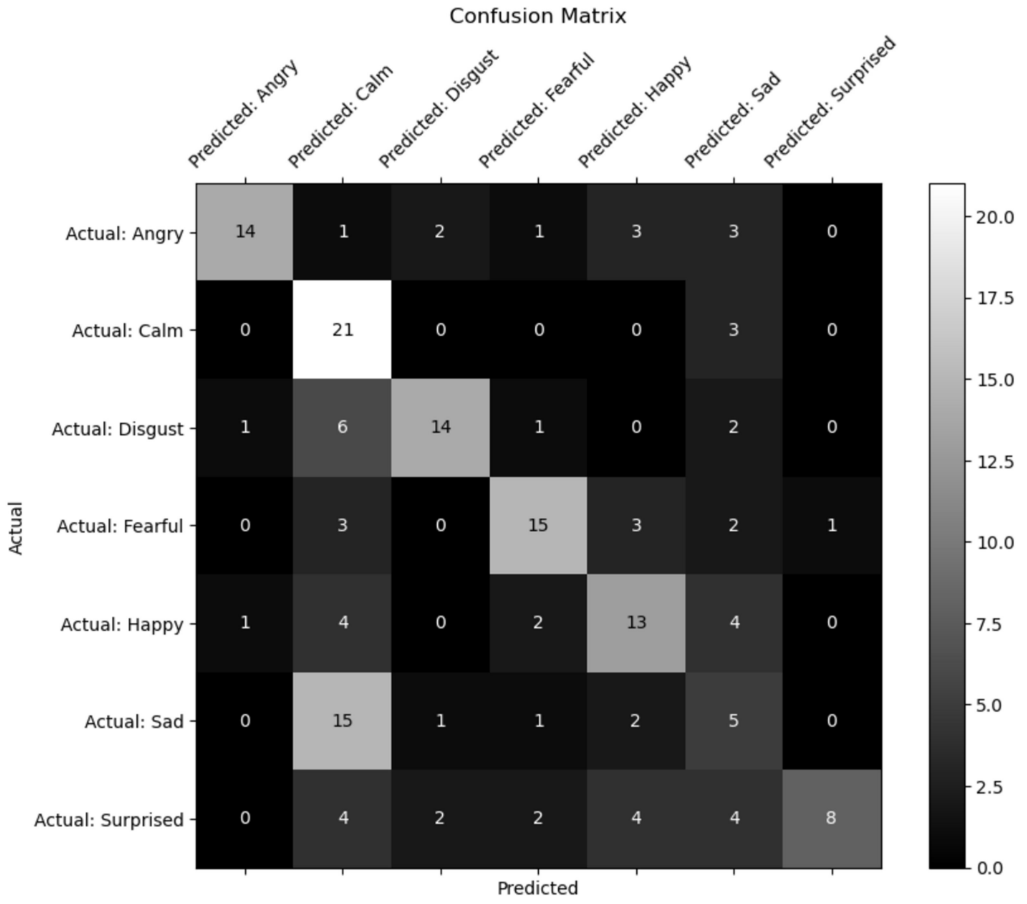
This was a method I came up with to extract features with Librosa into a dataframe then process the dataframe for later analysis. I implemented a system to eliminate data leakage that was discovered part way through this project. This significantly impacted the results by over 20% accuracy. After extensive hyper-parameter testing, I ran the classes through a pipeline to scale and use MLPClassifier with the best parameters I found and these were the results.

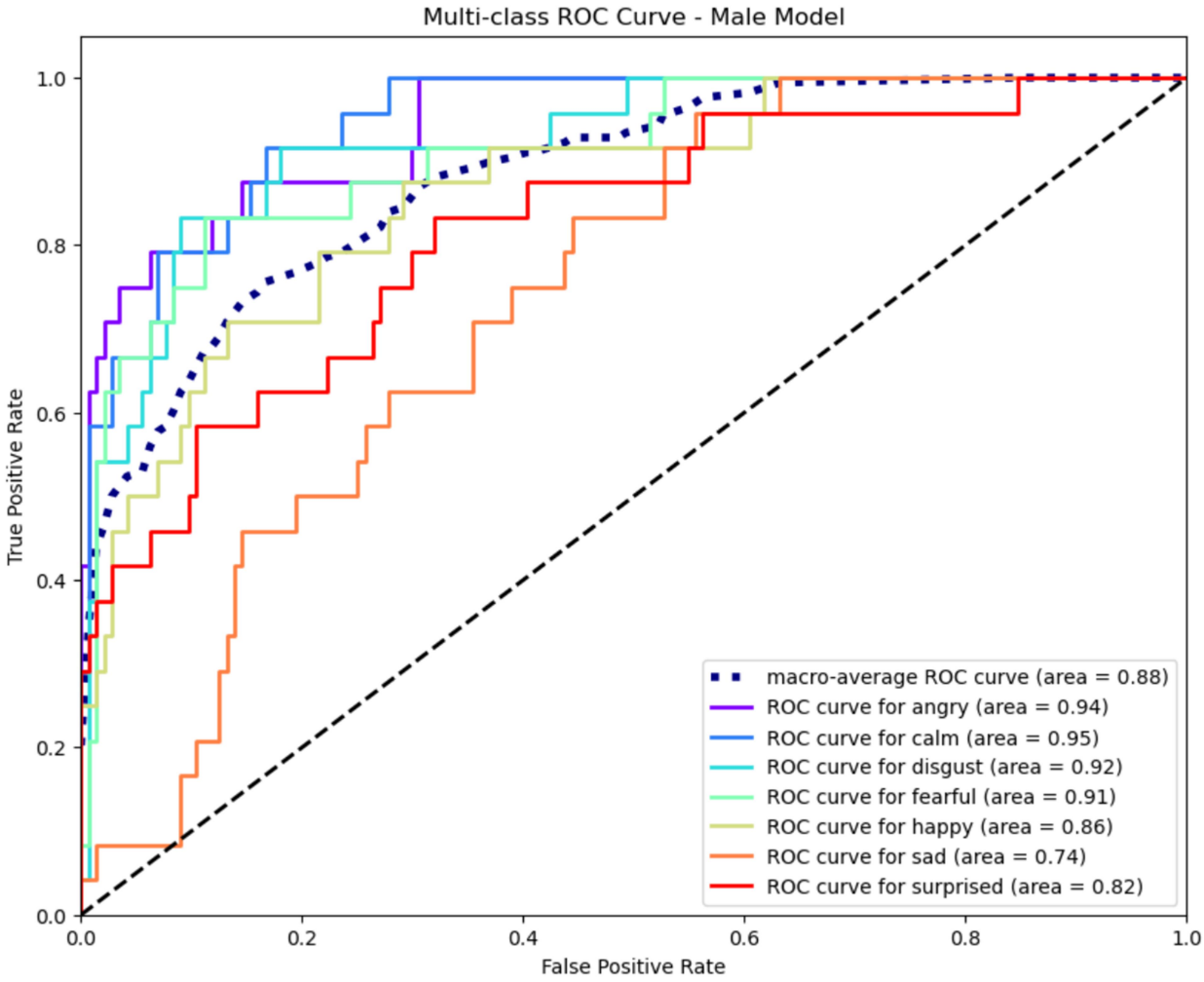
# Male dataset New Method

```
[[14  1  2  1  3  3  0]
 [ 0 21  0  0  0  3  0]
 [ 1  6 14  1  0  2  0]
 [ 0  3  0 15  3  2  1]
 [ 1  4  0  2 13  4  0]
 [ 0 15  1  1  2  5  0]
 [ 0  4  2  2  4  4  8]]
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| angry | 0.88 | 0.58 | 0.70 | 24 |
| calm | 0.39 | 0.88 | 0.54 | 24 |
| disgust | 0.74 | 0.58 | 0.65 | 24 |
| fearful | 0.68 | 0.62 | 0.65 | 24 |
| happy | 0.52 | 0.54 | 0.53 | 24 |
| sad | 0.22 | 0.21 | 0.21 | 24 |
| surprised | 0.89 | 0.33 | 0.48 | 24 |
| accuracy | | | 0.54 | 168 |
| macro avg | 0.62 | 0.54 | 0.54 | 168 |
| weighted avg | 0.62 | 0.54 | 0.54 | 168 |

Accuracy: 0.5357142857142857




Multi-class ROC Curve - Male Model
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
macro-average ROC curve (area = 0.88)
ROC curve for angry (area = 0.94)
ROC curve for calm (area = 0.95)
ROC curve for disgust (area = 0.92)
ROC curve for fearful (area = 0.91)
ROC curve for happy (area = 0.86)
ROC curve for sad (area = 0.74)
ROC curve for surprised (area = 0.82)

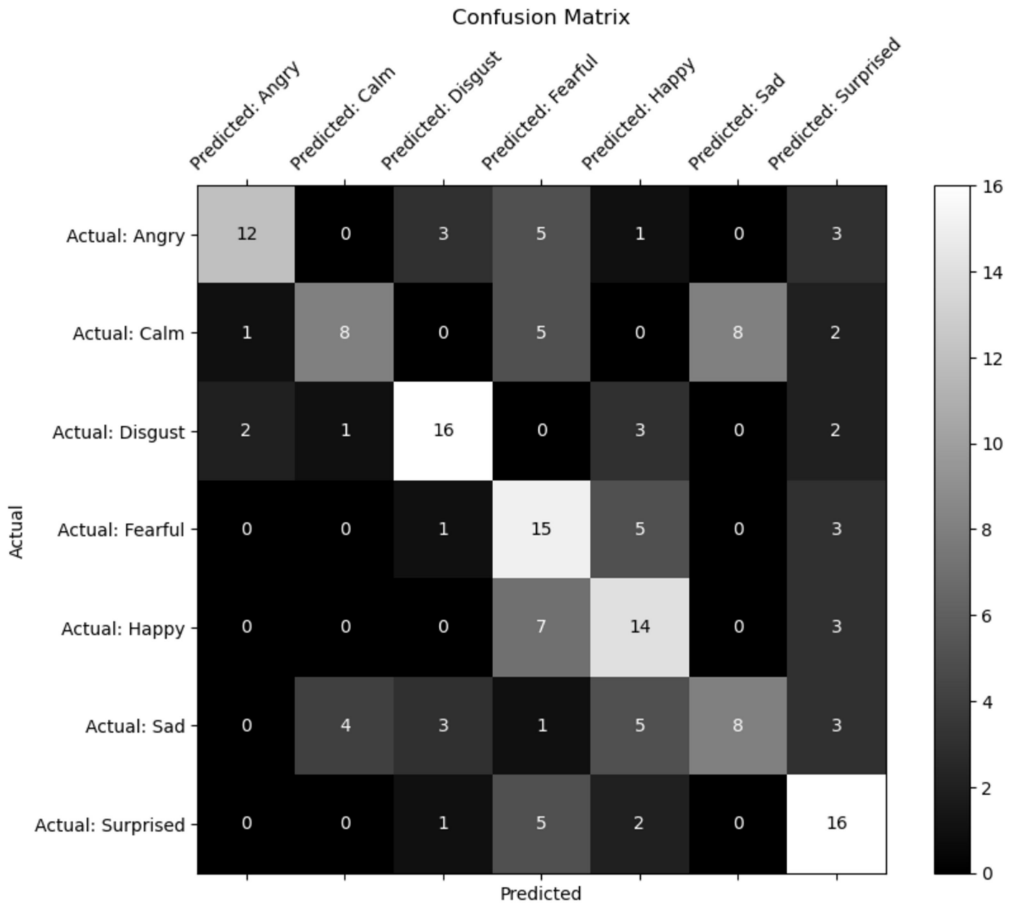
# Female Dataset New Method

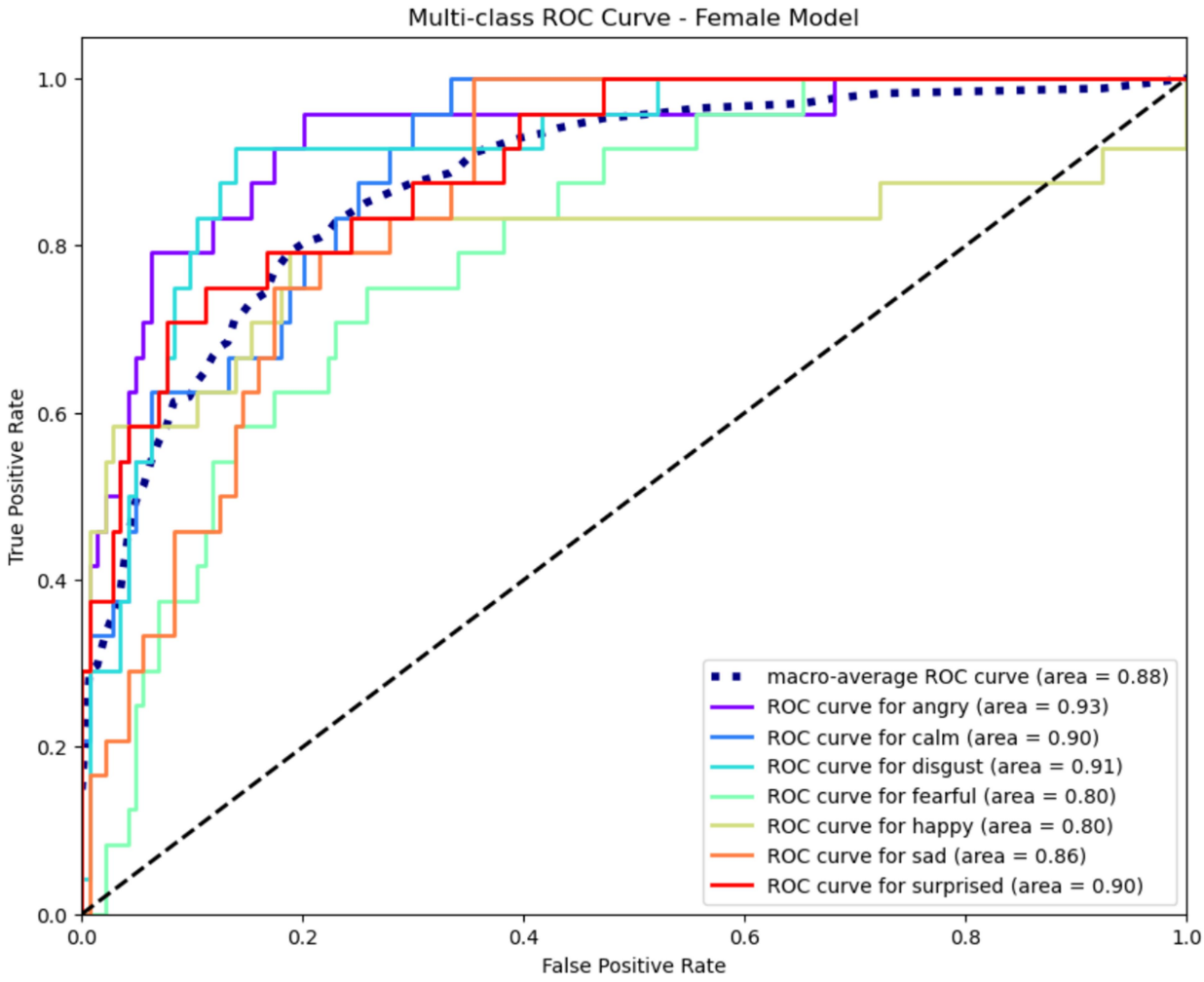
```
[[12  0  3  5  1  0  3]
 [ 1  8  0  5  0  8  2]
 [ 2  1 16  0  3  0  2]
 [ 0  0  1 15  5  0  3]
 [ 0  0  0  7 14  0  3]
 [ 0  4  3  1  5  8  3]
 [ 0  0  1  5  2  0 16]]
              precision    recall  f1-score   support

       angry       0.80      0.50      0.62        24
        calm       0.62      0.33      0.43        24
     disgust       0.67      0.67      0.67        24
     fearful       0.39      0.62      0.48        24
       happy       0.47      0.58      0.52        24
         sad       0.50      0.33      0.40        24
   surprised       0.50      0.67      0.57        24

    accuracy                           0.53       168
   macro avg       0.56      0.53      0.53       168
weighted avg       0.56      0.53      0.53       168

Accuracy: 0.5297619047619048
```






## Final Thoughts

After discovering and correcting the data leak, the results dropped significantly and I was unable to get them back up to an accuracy I was happy with. I did another notebook with this final process but with 4 classes of emotion instead (angry, calm, happy, sad) in order to target specific emotions that would apply to the objective of a call center scenario. This did improve the accuracy by about 10%, which is good but not very good. 63% to 65% isn't bad but I was hoping for somewhere above 80%. I did see another article that used Librosa to extract the spectrograph images and use vision as a method to classify emotion and I wonder if that would be the direction this field would be best to take.

Academic citation

Livingstone SR, Russo FA (2018) The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLoS ONE 13(5): e0196391. https://doi.org/10.1371/journal.pone.0196391.

All other attributions



Data-flair.training citation

https://data-flair.training/blogs/python-mini-project-speech-emotion-recognition/