AMATH 582: Emotion Classification
(Final Project)

Hyun Ah Lim
March 18, 2020

Abstract

Speech audio clips from the public dataset RAVDESS (ref [1]) were used to develop multiple emotion recognition models using Singular Value Decomposition (for Principal Component Analysis) and three supervised learning classification methods. Within each classification method, three approaches were attempted:

1. Use cumulative energy thresholds for mode selection to encompass the maximum variation in data within a minimum number of dimensions.
2. Split the dataset to remove an assumed variation in data that could cloud emotion recognition, the gender/sex of the voice. Execute cumulative energy mode selection in the constrained dataset and see if this improves accuracy.
3. Identify and employ high performing modes, regardless of their variation dominance. Use these modes rather than the highest energy modes (cumulative energy threshold approach described above).

The first two approaches produced similar results in terms of accuracy, where at 90% of the cumulative energy modes, the accuracy levels generally hovered between 15-30%. Limiting the dataset, approach #2, for male versus female clips did not improve overall accuracy. We can therefore assume that we cannot improve accuracy given the same methodology, by first finding an accurate way to classify male vs female voices. Both approaches, however, did show some promise in terms of efficiency in that accuracy generally recorded above random chance while reducing the dimensionality, 671 used out of 1440 total modes (90% of total cumulative energy).
The third approach, proved to be extremely accurate for a limited number of emotions, however, it did not produce consistent results across all emotion classes, performing below random chance in some. It is important to note that in this case only eight modes were used in each classification group, much smaller than the number of modes used in approaches #1 and #2 (maximum of 671). Therefore, based on the success for specific emotion classes and on the efficiency of using only a small number of modes, I think this approach warrants further investigation to determine if a specific combination of modes (regardless of their singular values) could provide an extremely efficient and accurate model.

1 Introduction and Overview

This analysis will combine the data transformation techniques of Fast Fourier Transform (FFT) and Singular Value Decomposition (SVD) for Principal Component Analysis (PCA). Three supervised learning classification methods will then be used on this transformed data to provide emotion recognition capability. The speech audio clips from the public RAVDESS dataset (ref [1]) described below are used. Each audio clip is approximately 3 seconds in duration and contains the audio of an actor speaking in one of the eight emotion categories (or classes) listed below:

01- Neutral	02- Calm	03- Happy	04- Sad
05- Calm	06- Fearful	07- Disgust	08- Surprised

There are 24 different actors, of which 12 are male and 12 are female. Each actor performs the statement with the given emotion in two different intensity levels. with the exception of the neutral emotion. Each of these is then repeated to produce two different clips from each actor with the same emotion and intensity. This brings the dataset total to 1440 audio clips. The frequency content of each of these will be transformed using SVD and the results used to classify 'new' clips of music within each dataset. The classification methods applied in this exercise will be Linear Discrimination Analysis, Naïve Bayes and K Nearest Neighbor, all supervised learning methods.

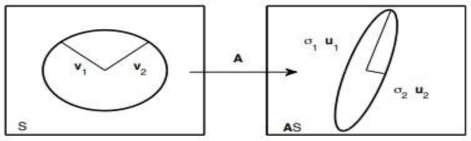
To explore the effectiveness of applying SVD to frequency content with each of the classification methods, for emotion recognition purposes, three different approaches will be assessed:

- Cumulative energy thresholds will be used to select the modes (orthogonal bases) that contain the highest variations (i.e. use modes with highest Singular Values). These modes will be used for classification purposes.
- Remove an assumed/perceived dominant feature space, gender of the voice, and employ the classification methodology within each dataset (male vs. female).
- Identify specific high performing modes by emotion and classification method, even if they correspond to a lower singular value than others. Then use this group of high performing modes by classification method for overall accuracy across all emotion classes.

2 Theoretical Background

This analysis employs Singular Value Decomposition (SVD) as a means to transform the data in order to classify each speech clip into one of eight distinct emotion classes. This data transformation reduces dimensionality, while capturing the maximum variance for a given dimension/feature space. We will be using the dominant feature spaces (Principal Component Analysis, PCA) as one approach for classification but also attempting a second that tries to identify single features (i.e. dimensions/modes) that perform well within a given classification method. Finally, we will remove a perceived dimension (variation in frequency by gender) and to explore whether that improves classification accuracy.

Singular Value Decomposition (for Principal Component Analysis) has been discussed in previous papers in this class series, but as a quick review, SVD decomposes any matrix into its basic actions, rotation and scaling. An illustration of what happens to vectors v1, and v2 are shown below in figure 1.



$$A\,V = U\,\Sigma$$

FIGURE 1. EXAMPLE WHERE VECTORS, V1 AND V2, ARE HIT WITH MATRIX A, AND THE RESULTING IMPACT THE EQUATION THAT REPRESENTS THIS IS SHOWN IN EQUATION FORM AS WELL AS MATRIX FORM (REF[2]).

You can then rewrite this to outline the SVD components of any matrix A: U, Σ and V.

$$A = U\,\Sigma\,V^T \qquad \text{(eq 1)}$$

The U, Σ and V elements described below:

- U: The columns of this matrix are the orthonormal bases, ordered by the level of variation of the dataset in that basis. They are the eigenvectors of the covariance matrix of A (AA^T). They represent the feature dimensions that we can use to define matrix A, ordered by dominance (i.e. energy, variance, etc.).
- Σ: These are the singular values that tell us the level of dominance in each of the corresponding orthogonal bases.
- V: This matrix contains the 'projection' information for each column of A into each orthogonal base. Each row of V represents a single column within the original matrix A. In other words, each V contains the discrete information of each audio clip and how it projects into each of the dimensions defined in U.

For the classification of audio in this study, SVD will be performed on the Fourier Transformed frequency content of each clip (discussed in a previous paper in this series). The V output matrix from SVD contains the discriminating elements between the audio clips and will be used for data classification purposes.

The first approach, during classification, will use cumulative energy thresholds, capturing maximum variation within a minimum number of dimensions. In other words, the dimensions with the highest variation, will be used to accurately classify the clips into the eight separate emotion classes. Cumulative energy (as calculated using the Singular Values) will be assessed at intervals and those modes will be evaluated for accuracy. If this approach provides positive results at relatively low cumulative energy thresholds then, Principal Components, would be an effective approach to emotion recognition.

A second approach will split the data into two separate groups, by gender, in attempts to remove one perceived dominant variation, the difference in male vs. female voices. If the accuracy significantly increases over the first approach when the entire data set is used, then classification by gender prior to emotion classification could be an effective methodology to increase accuracy.

The third approach considers the possibility that bases/modes with lower variation (lower Singular values) may play an important role in emotion detection. Therefore, additional analysis will be performed to identify modes that that may not be dominant but provide high accuracy for the three classification methods outlined below.

In each of these approaches, the three supervised machine learning classification methods described below will be used :

<u>Linear Discrimination Analysis</u> works to find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data. In other words, construct a projection such that the equation below is true (ref [2]). This discriminate line for projection is determined using the training data.

$$\mathrm{w} = \mathrm{argmax}_{\mathrm{W}} \frac{w^T\ \mathrm{s}_\beta w}{w^T\ \mathrm{s}_\mathrm{w} w} \qquad \text{(eq 2)}$$

<u>Naïve Bayes</u> method takes a statistical approach by using the probability of an event occurring in the prediction. If the prediction is binary, then this can be written as (ref [2]):

$$\frac{P(1|x)}{P(0|x)} = \frac{P(1|x) * P(1)}{P(0|x) * P(0)} \qquad \text{(eq 3)}$$

If you know the ratio ($\frac{P(1|x)}{P(0|x)}$) then you can make a prediction (the training data is used to generate this ratio)

<u>K Nearest Neighbor</u> predicts the classification by searching for the n number of nearest neighbors (in the training set and their classifications) to predict the test data point. 'Nearest' is in terms of distance (eq 4) to the nearest neighbor(s) within the features space (dimensions) selected.

All three supervised learning methods utilize a dataset that includes labels and the classification model is trained using this known information. The test data set will comprise approximately 80% of the data, and testing will be performed on the remaining 20%. This will then randomly be reshuffled and iterated many times for cross validation purposes and to calculate mean accuracy percentages.

3 Algorithm Implementation and Development

The algorithm was developed in using four separate scripts, executed in a similar and are outlined below:

3.1 Classification with cumulative energy thresholds for mode selection (Script 1, part 1):

3.1.1 Load and compile matrix of all audio clips

Use dir() function to generate a list of the filenames and locations by emotion category. Use this list to execute a for loop, where each iteration imports a single file, reshapes it to a column vector, takes the discrete fast Fourier transform, shifts the values so zero is in the center, fftshift(), and then appends it to a single matrix. Repeat this step for each emotion.

3.1.2 Concatenate all matrices horizontally, using horzcat(), and normalize by subtracting from each element the mean value of its row.

3.1.3 Execute SVD to generate matrices, U, Σ and V.

3.1.4 Extract Singular Values from the diagonal matrix, Σ, to determine dominance of each mode, and plot.

3.1.5 Use a for loop to establish the minimum mode to reach cumulative energy increment totals of 10 (i.e. 10%, 20%, 30%, etc.). Set this up as a table for reference later.

3.1.6 Use for loop to perform the following Classification algorithm for 1:n modes that comprise 30, 40, 50, 60, 70, 80 and 90% of the cumulative energy as determined from step 3.1.5

3.1.6.1 Use inner for loop to execute 50 iterations (for cross validation) for Linear Discrimination Classification. This involves randomly selecting 152 clips from each emotion class to train, and the balance of 40 to test, using the MATLAB classify() function. Each of the training clips is provided a class value of 1-8, depending on the emotion class it belongs. The predicted class value of the remaining test clips are then compared to the actual class to which it belongs. In this case, the V matrix from the SVD output is used as the input representing the emotion expressed in the clip.

3.1.6.2 Calculate the mean accuracy for each emotion class/cumulative energy threshold and plot for visual comparison.

3.1.6.3 Repeat steps 3.1.6.1-3.1.6.2 using classification methods Naïve Bayes and K Nearest Neighbor.

3.2 Classification within Male Actor Clips, cumulative energy mode selection (Script 2)

3.2.1 Repeat steps 3.1.1 – 3.1.6, with the exception of excluding female actor audio clips (this will reduce the total dataset, and therefore training and test datasets by half). This was performed to test whether the difference in female vs. male voices was one of the causes of inaccuracy in Script 1. Results did not prove more accurate than described in 3.1.

3.3 Classification within Female Actor Clips, cumulative energy mode selection (Script 3)

3.3.1 Repeat steps 3.1.1 – 3.1.6, with the exception of excluding male actor audio clips (this will reduce the total dataset, and therefore training and test datasets by half. Results did not prove more accurate than described in 3.1.

3.4 Classification using high performing mode selection (Script 1, part 2)

3.4.1 Use a for loop to execute the classification algorithm described in 3.1.6 above, using only a single mode rather than cumulative energy thresholds. Perform this for all 1440 modes for each of the three classification methods (Linear Discrimination Analysis, Naïve Bayes and K Nearest Neighbor). 50 iterations performed for each of these combinations for cross validation purposes.

3.4.2 Then the mean accuracies for each emotion for each mode/classification method was plotted to see if any single mode classification combination visually appeared to have significantly better results from the algorithm of 3.1.6. From this plot, it's apparent that specific mode/classification combinations perform well for specific classes (emotions) but a single combination did not perform significantly better across all classifications (emotions).

3.4.3 Based on accuracy results, identify, if any exist, 'high performing' modes, both by emotion class and by classification method. The next part will use this information to select a combination of modes for a single classification method to see if this improves the overall accuracy.

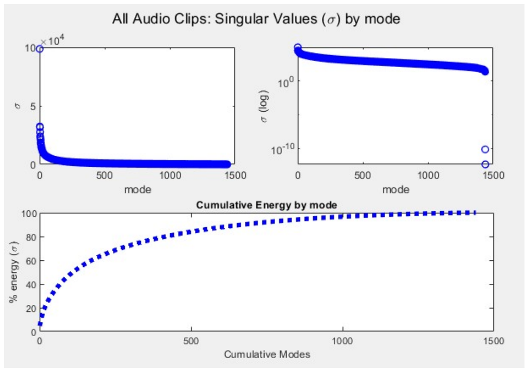
3.4.4 Compile matrix that gathers the highest performing mode for each emotion and classification method (i.e. for a given classification method, select the one mode for each emotion that provides the highest mean accuracy, for a total of eight modes).

3.4.5 Repeat the algorithm described in steps 3.1.6.1-3.1.6.2 using the high performing modes for each of the classification methods (different modes for the different classification methods). Plot to assess if this method increased accuracy across all emotion classes.

4 Computation Results

4.1 Sigma Values (All Clips, Male Audio Clips, Female Audio Clips)

Below are the Singular (σ) Values by mode, for the three datasets (figure 2): all audio clips (720 male, 720 female), male audio only (720) and female audio only (720).



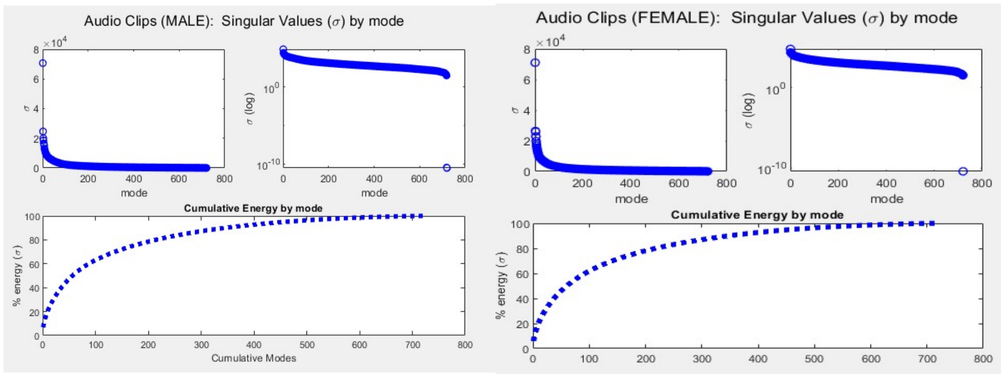


FIGURE 1. SINGULAR VALUES SHOWS FOR ALL AUDIO CLIPS (TOP ROW), MALE AUDIO CLIPS ONLY (SECOND ROW LEFT) AND FEMALE AUDIO CLIPS ONLY (SECOND ROW RIGHT).

4.2 Classification utilizing the V element of SVD.

4.2.1 All Clips, Cumulative Energy

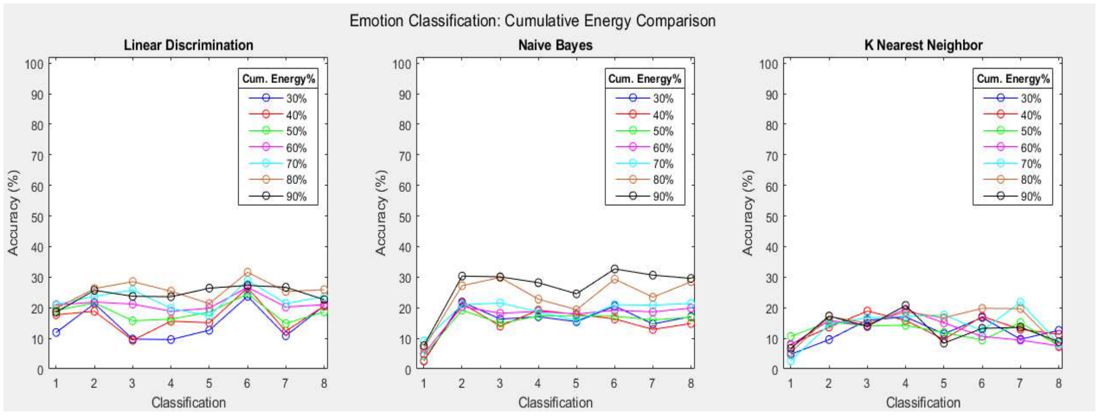


FIGURE 1. ACCURACY RESULTS FOR EACH CLASSIFICATION METHOD IS PLOTTED FOR EACH OF THE MODE GROUPS (BY CUMULATIVE ENERGY THRESHOLDS, 30-90%). RESULTS ARE BASED ON USAGE OF ENTIRE DATASET.

4.2.2 Male Audio Clips, classification by cumulative energy modes

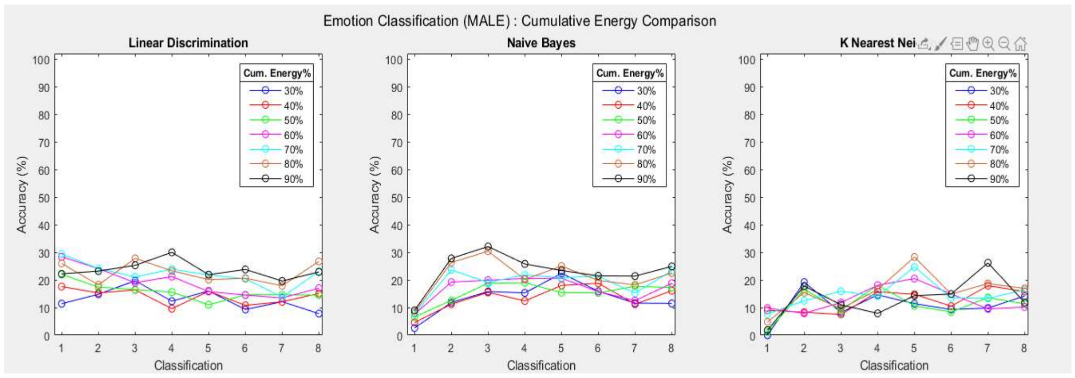


FIGURE 1. ACCURACY RESULTS FOR EACH CLASSIFICATION METHOD IS PLOTTED FOR EACH OF THE MODE GROUPS (BY CUMULATIVE ENERGY THRESHOLDS, 30-90%). RESULTS ARE BASED ON DATA SUBSET (MALE ONLY).

4.2.3 Female Audio Clips, classification by cumulative energy modes

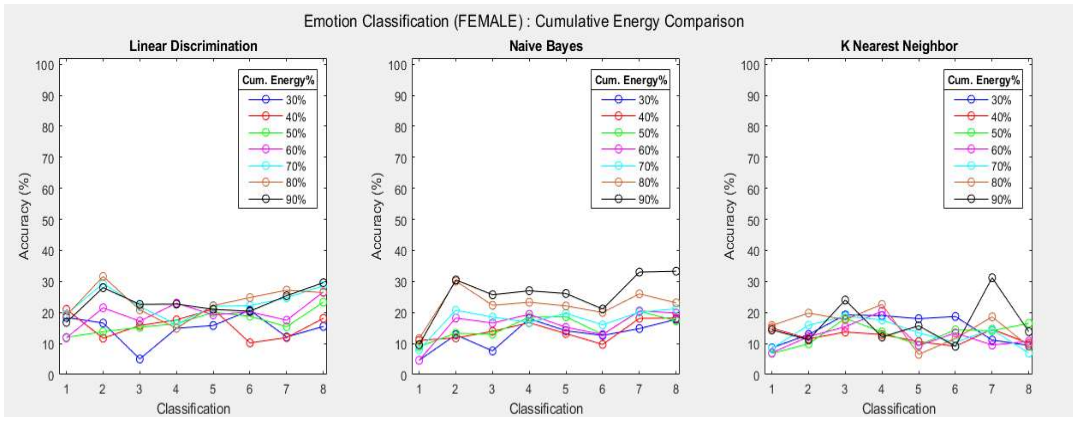


FIGURE 1. ACCURACY RESULTS FOR EACH CLASSIFICATION METHOD IS PLOTTED FOR EACH OF THE MODE GROUPS (BY CUMULATIVE ENERGY THRESHOLDS, 30-90%). RESULTS ARE BASED ON DATA SUBSET (FEMALE ONLY).

4.2.4 All Clips, High Performing Mode Selection

4.2.4.1 Identification of highest performing mode in each classification method for each emotion.

After running the algorithm for all three classification methods against each single mode, the following were identified with the highest mean accuracy given 50 iterations.

TABLE 1: LIST OF MODES THAT PERFORMED WITH THE HIGHEST MEAN ACCURACY BY CLASSIFICATION METHOD AND CLASSIFICATION LABEL (EMOTION).

Classification	Emotion	Linear Discrimination		Naïve Bayes		K Nearest Neighbor	
		mode	accuracy	mode	accuracy	mode	accuracy
01	Neutral	33	61%	1439	27%	1440	25%
02	Calm	1328	58%	1440	92%	1440	60%
03	Happy	129	56%	526	60%	403	24%
04	Sad	637	57%	873	63%	257	25%
05	Angry	1014	58%	1124	59%	1325	21%
06	Fearful	1102	58%	1071	61%	995	22%
07	Disgust	922	58%	1439	79%	1439	27%
08	Surprised	1426	58%	501	62%	469	22%

4.2.4.2 Classification using modes identified as 'high performers'

Each of the three classification methods was used with their corresponding eight highest performing modes listed in 4.2.4.1. The results are shown below in figure x. Although there are specific emotions with significant increase in accuracy (versus the cumulative energy modes), we see that the improvement is not consistent across all emotion classes for a single classification method.

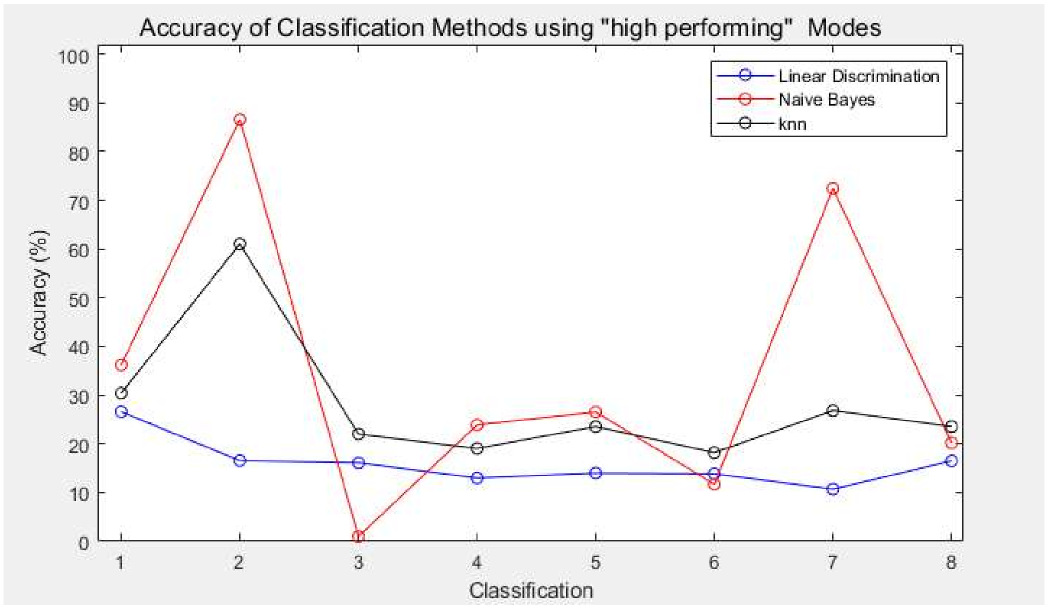


FIGURE 6: THE AVERAGE ACCURACY OVER 50 ITERATIONS FOR EACH CLASSIFICATION METHOD USING HIGHEST PERFORMING MODES FOR EACH EMOTION CLASS ARE SUMMARIZED ABOVE.

5 Summary and Conclusions

This study attempted to develop an emotion classification model using three supervised learning classification methods: Linear Discrimination Analysis (LD), Naïve Bayes (NB) and K Nearest Neighbor (KNN). The data were first transformed into their frequency components using the Fast Fourier Transform method, and then a Singular Value Decomposition was performed to allow for a Principal Component Analysis. These data were then used to train and test each of the models.

In each model three approaches were taken:

- Cumulative energy thresholds mode selection was used to encompass the maximum variation in data within a minimum number of dimensions. Both LD and NB provided somewhat consistent results higher than if the classification was done randomly (1/8=12.5%), with the exception of the neutral emotion. In addition, both classification methods seemed to improve at higher cumulative energy thresholds. K Nearest Neighbor seemed to be the method that did not show consistent improvement with increasing cumulative energy. Note that at the highest cumulative energy threshold of 90% , 671 modes out of the total 1440 were required.

- When removing an assumed/perceived dominant feature space, gender of the voice, by limiting the training and test data set within a single gender did not provide significant improvement in accuracy. Therefore, a possible two step classification method where gender is first classified and would not improve accuracy given the current method.
- Identifying single modes that were the highest performers for specific classification method and emotion class combinations, did significantly improve the classification accuracy for some of the emotion classes but did not provide a consistent or significant improvement in all classes. However, this only utilized 8 out of the 1440 total modes and therefore was extremely efficient for those successful predictions.

Although variation based on emotion was the feature targeted in this study, additional variations based on individual voice (24 actors) and gender (male and female actors) and intensity may have contributed to the overall accuracy issues found in all three approaches. Based on the success of some of the emotion classes/classification methods using the high performing modes approach, further exploration of different combinations of modes (at all Singular value levels), may result in identification of a 'high performing group' that significantly provides more accurate and consistent results, while also proving extremely efficient.

References

[1] 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.

[2] J. Nathan Kutz 2013, Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, Oxford University Press

Appendix A: Relevant MATLAB functions

1. dir(): Provides a list of the filenames and locations in a given path. This was used to import each image/audio clip in the respective datasets.
2. horzcat(): Horizontally concatenates. This was used to stack the truncated x and y vectors to build the A matrix.
3. uint64(): converts value to unsigned 64-bit
4. svd(): Performs SVD on a given matrix, outputting matrices [U,S, V].
5. diag(): converts a diagonal matrix (extracting the non-zero values) to a column vector
6. find(): find the index values of a specific non-zero value in an array
7. audioread(): reads audio files as two outputs, Y (sampled data) and FS (sample rate in Hertz)
8. randperm(): generates an N length vector containing a random permutation of the integers 1:N
9. classify(): function used to perform the Linear Discrimination Analysis (supervised learning)
10. fitcnb(): fits a Naïve Bayes classifier to data
11. nb.predict(): function used to perform Naïve Bayes classification (supervised learning)
12. knnsearch(): function used to perform K Nearest Neighbor classification (supervised learning)
13. repmat(): replicate and tile and array. This was used in normalization calculations for a matrix
14. nnz(): computes number of non-zero elements in an array
15. ind2sub(): provides multiple subscripts from a given linear index value, given input matrix size
16. 'for loop' (honorable mention): This isn't a function but the 'for loop' allowed us to perform iterations across all 20 measurements with minimal coding.

Appendix B: Github link

https://github.com/Washington-Turtles/amath582.git

Appendix C: MATLAB Code

Script 1 (emotion.m)

```
% load files, truncate, sort male vs. female, fourier transform
% Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
clear all; close all; clc;
n = 140900;
% emotion = neutral
pgmfiles=dir('*/*/03-01-01-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y1, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y1 = y1(1:n);
   y1_all(:, ii) = y1;
   v1 = y1'; y1f = fft(v1); y1fs = abs(fftshift(y1f));
   y1fs_all(:, ii) = y1fs;
end
% emotion = calm
pgmfiles=dir('*/*/03-01-02-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y2, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y2 = y2(1:n);
   y2_all(:, ii) = y2;
   v2 = y2'; y2f = fft(v2); y2fs = abs(fftshift(y2f));
   y2fs_all(:, ii) = y2fs;
end
% emotion = happy
pgmfiles=dir('*/*/03-01-03-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y3, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y3 = y3(1:n);
   y3_all(:, ii) = y3;
   v3 = y3'; y3f = fft(v3); y3fs = abs(fftshift(y3f));
   y3fs_all(:, ii) = y3fs;
end
% emotion = sad
pgmfiles=dir('*/*/03-01-04-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y4, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y4 = y4(1:n);
   y4_all(:, ii) = y4;
   v4 = y4'; y4f = fft(v4); y4fs = abs(fftshift(y4f));
   y4fs_all(:, ii) = y4fs;
end
% emotion = angry
pgmfiles=dir('*/*/03-01-05-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y5, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y5= y5(1:n);
   y5_all(:, ii) = y5;
   v5 = y5'; y5f = fft(v5); y5fs = abs(fftshift(y5f));
   y5fs_all(:, ii) = y5fs;
end
% emotion = fearful
```

```
pgmfiles=dir('*/*/03-01-06-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y6, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y6= y6(1:n);
    y6_all(:, ii) = y6;
    v6 = y6'; y6f = fft(v6); y6fs = abs(fftshift(y6f));
    y6fs_all(:, ii) = y6fs;
end
% emotion = disgust
pgmfiles=dir('*/*/03-01-07-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y7, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y7= y7(1:n);
    y7_all(:, ii) = y7;
    v7 = y7'; y7f = fft(v7); y7fs = abs(fftshift(y7f));
    y7fs_all(:, ii) = y7fs;
end
% emotion = surprise
pgmfiles=dir('*/*/03-01-08-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y8, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y8= y8(1:n);
    y8_all(:, ii) = y8;
    v8 = y8'; y8f = fft(v8); y8fs = abs(fftshift(y8f));
    y8fs_all(:, ii) = y8fs;
end
y_all = horzcat(y1_all, y2_all, y3_all, y4_all, y5_all, y6_all, y7_all, y8_all);
yfs_all = horzcat(y1fs_all, y2fs_all, y3fs_all, y4fs_all, y5fs_all, y6fs_all, y7fs_all, y8fs_all);

%  normalize (take mean along rows) and perform SVD to find singular values
[m,n] = size(yfs_all);
mn=mean(yfs_all,2); %compute mean for each row
clips_mean=yfs_all -repmat(mn, 1, n); % subtract mean
[u,s,v] = svd(clips_mean', 'econ');
 sig = diag(s);
energy_all = zeros(n, 2);

%calculate cumulative energy
for ee = 1:n
    energy_s = sum(sig(1:ee))/sum(sig) ;
    energy_all(ee, 2) = energy_s;
    energy_all(ee, 1) = ee;
end
%%
figure(1)
 sgtitle('All Audio Clips: Singular Values (\sigma) by mode')
 subplot(2,2,1)
plot(sig, 'bo', 'Linewidth', [1]);
xlabel('mode')
ylabel('\sigma')
subplot(2,2,2)
semilogy(sig,'bo','Linewidth',[1])
xlabel('mode')
ylabel('\sigma (log)')
```

```
subplot(2,1, 2)
plot(energy_all(:, 2)*100, 'b:', 'Linewidth', [4])
title('Cumulative Energy by mode')
ylabel('% energy (\sigma)')
xlabel('Cumulative Modes')

% calculate table with cum energy at 10% segment  thresholds
energy_summ = zeros(10, 2);
for ee = 0.1:0.1: 1
thres = find(energy_all(:,2) >= ee);
min_loc = uint64(min(thres));
energy_summ(uint64(ee*10), :) = energy_all(min_loc, :);
end
s_table = table(energy_summ(:, 1), energy_summ(:, 2), 'VariableNames', {'Mode'; 'Energy %'});

%% Supervised Learning
% Cross valudation using 100 iter of sampled train vs test data.
m_vector = [30:10:90];
for mm = m_vector
modes = min(find(energy_all(:, 2) >mm/100));
for jj = 1:50
q1 = randperm(96);
q2 = randperm(file_count);
q3 = randperm(file_count);
q4 = randperm(file_count);
q5 = randperm(file_count);
q6 = randperm(file_count);
q7 = randperm(file_count);
q8 = randperm(file_count);
oo = 96;% neutral
x1start = 1; x1end = file_count-oo;
x1= v(x1start : x1end, 1:modes);
x2start = file_count+1-oo; x2end =(file_count*2)-oo;
x2 = v(x2start : x2end, 1:modes);
x3start = (2*file_count)+ 1-oo; x3end = (file_count*3)-oo;
x3 = v(x3start : x3end, 1:modes);
x4start = (3*file_count)+ 1-oo;  x4end =(file_count*4)-oo;
x4 = v(x4start : x4end, 1:modes);
x5start = (4*file_count)+ 1-oo;  x5end =(file_count*5)-oo;
x5 = v(x5start : x5end, 1:modes);
x6start = (5*file_count)+ 1-oo; x6end =  (file_count*6)-oo;
x6 = v(x6start : x6end, 1:modes);
x7start = (6*file_count)+ 1-oo; x7end = (file_count*7)-oo;
x7 = v(x7start : x7end, 1:modes);
x8start = (7*file_count)+ 1-oo; x8end = (file_count*8)-oo;
x8 = v(x8start : x8end, 1:modes);
% setup train/test datasets
ntr = 76;
tr = 152;
xtrain = [x1(q1(1:ntr), :); x2(q2(1:tr), :); x3(q3(1:tr), :); x4(q4(1:tr), :); x5(q5(1:tr), :); x6(q6(1:tr), :); x7(q7(1:tr), :);
x8(q8(1:tr), :)];
xtest = [x1(q1(ntr+1:end), :); x2(q2(tr+1:end), :); x3(q3(tr+1:end), :); x4(q4(tr+1:end), :); x5(q5(tr+1:end), :);
x6(q6(tr+1:end), :); x7(q7(tr+1:end), :); x8(q8(tr+1:end), :)];
ctrain = [ones(ntr, 1); 2*ones(tr, 1); 3*ones(tr, 1); 4*ones(tr, 1); 5*ones(tr, 1); 6*ones(tr, 1); 7*ones(tr, 1); 8*ones(tr, 1)];
```

```
% linear discrimination analysis
pre1 = classify(xtest, xtrain, ctrain);
exp_class = [ones(20, 1); 2*ones(40, 1); 3*ones(40, 1); 4*ones(40, 1); 5*ones(40, 1); 6*ones(40, 1); 7*ones(40, 1);
8*ones(40, 1)];
error1 = nnz(pre1(1:20) - 1);
accur1 = (20-error1)/20;
error2 = nnz(pre1(21:60) - 2);
accur2 = (40-error2)/40;
error3 = nnz(pre1(61:100) - 3);
accur3 = (40-error3)/40 ;
error4 = nnz(pre1(101:140) - 4);
accur4 = (40-error4)/40 ;
error5 = nnz(pre1(141:180) - 5);
accur5 = (40-error5)/40 ;
error6 = nnz(pre1(181:220) - 6);
accur6 = (40-error6)/40 ;
error7 = nnz(pre1(221:260) - 7);
accur7 = (40-error7)/40 ;
error8 = nnz(pre1(261:300) - 8);
accur8 = (40-error8)/40 ;

ld_accur(jj, :)  = horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
ld_mean_accuracy = mean(ld_accur, 1);
% naive bayes
nb = fitcnb(xtrain, ctrain);
pre2 = nb.predict(xtest);

error1 = nnz(pre2(1:20) - 1);
accur1 = (20-error1)/20;
error2 = nnz(pre2(21:60) - 2);
accur2 = (40-error2)/40;
error3 = nnz(pre2(61:100) - 3);
accur3 = (40-error3)/40 ;
error4 = nnz(pre2(101:140) - 4);
accur4 = (40-error4)/40 ;
error5 = nnz(pre2(141:180) - 5);
accur5 = (40-error5)/40 ;
error6 = nnz(pre2(181:220) - 6);
accur6 = (40-error6)/40 ;
error7 = nnz(pre2(221:260) - 7);
accur7 = (40-error7)/40 ;
error8 = nnz(pre2(261:300) - 8);
accur8 = (40-error8)/40 ;

nb_accur(jj, :)  =  horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
nb_mean_accuracy = mean(nb_accur, 1);

% k nearest neighbor
[ind, D] = knnsearch(xtrain, xtest);

for ii = 1:length(ind)
   pre3(ii, :) = ctrain(ind(ii));
end
```

```
error1 = nnz(pre3(1:20) - 1);
accur1 = (20-error1)/20;
error2 = nnz(pre3(21:60) - 2);
accur2 = (40-error2)/40;
error3 = nnz(pre3(61:100) - 3);
accur3 = (40-error3)/40 ;
error4 = nnz(pre3(101:140) - 4);
accur4 = (40-error4)/40 ;
error5 = nnz(pre3(141:180) - 5);
accur5 = (40-error5)/40 ;
error6 = nnz(pre3(181:220) - 6);
accur6 = (40-error6)/40 ;
error7 = nnz(pre3(221:260) - 7);
accur7 = (40-error7)/40 ;
error8 = nnz(pre3(261:300) - 8);
accur8 = (40-error8)/40 ;

knn_accur(jj, :)  = horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
knn_mean_accuracy = mean(knn_accur, 1);

end
results((mm/10)-2, :) = [ld_mean_accuracy, nb_mean_accuracy,knn_mean_accuracy];
end

%%
figure(3)
sgtitle('Emotion Classification: Cumulative Energy Comparison')
x = 1:8;
subplot(1,3, 1)
plot(x, results(1, 1:8)*100, 'b-o'); hold on;
plot(x, results(2, 1:8)*100, 'r-o');
plot(x, results(3, 1:8)*100, 'g-o');
plot(x, results(4, 1:8)*100, 'm-o');
plot(x, results(5, 1:8)*100, 'c-o');
plot(x, results(6, 1:8)*100, '-o','Color',[.8,.4,.2]);
plot(x, results(7, 1:8)*100, 'k-o');
xlim([.8 8.1]); xlabel('Classification')
ylim([0 102])
title('Linear Discrimination')
legend( '30%','40%', '50%', '60%','70%', '80%','90%', 'Location', 'NorthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3,2)
plot(x, results(1,9:16)*100, 'b-o'); hold on;
plot(x, results(2, 9:16)*100, 'r-o');
plot(x, results(3, 9:16)*100, 'g-o');
plot(x, results(4, 9:16)*100, 'm-o');
plot(x, results(5, 9:16)*100, 'c-o');
plot(x, results(6, 9:16)*100,'-o','Color',[.8,.4,.2]);
plot(x, results(7, 9:16)*100, 'k-o');
xlim([.8 8.1]); xlabel('Classification')
ylim([0 102])
```

```
title('Naive Bayes')
legend( '30%','40%', '50%', '60%','70%', '80%','90%', 'Location', 'NorthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3, 3)
plot(x, results(1, 17:24)*100, 'b-o'); hold on;
plot(x, results(2, 17:24)*100, 'r-o');
plot(x, results(3,17:24)*100, 'g-o');
plot(x, results(4, 17:24)*100, 'm-o');
plot(x, results(5, 17:24)*100, 'c-o');
plot(x, results(6, 17:24)*100, '-o','Color',[.8,.4,.2]);
plot(x, results(7, 17:24)*100, 'k-o');
xlim([.8 8.1]); xlabel('Classification')
ylim([0 102])
title('K Nearest Neighbor')
legend( '30%','40%', '50%', '60%','70%', '80%','90%', 'Location', 'NorthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% single mode trial  %%%%%%%%%%%%%%%%%
%% Supervised Learning
% Cross valudation using 50 iter of sampled train vs test data.
%m_combos  = nchoosek(1:39, 38);
%for mm = 1:length(m_combos(:, 1))
for mm = 1:1440
%modes = m_combos(mm, :);
modes = mm;
for jj = 1:50
q1 = randperm(96);
q2 = randperm(file_count);
q3 = randperm(file_count);
q4 = randperm(file_count);
q5 = randperm(file_count);
q6 = randperm(file_count);
q7 = randperm(file_count);
q8 = randperm(file_count);
oo = 96;% neutral
x1start = 1; x1end = file_count-oo;
x1= v(x1start : x1end, modes);
x2start = file_count+1-oo; x2end =(file_count*2)-oo;
x2 = v(x2start : x2end, modes);
x3start = (2*file_count)+ 1-oo; x3end = (file_count*3)-oo;
x3 = v(x3start : x3end, modes);
x4start = (3*file_count)+ 1-oo;  x4end =(file_count*4)-oo;
x4 = v(x4start : x4end, modes);
x5start = (4*file_count)+ 1-oo;  x5end =(file_count*5)-oo;
x5 = v(x5start : x5end, modes);
x6start = (5*file_count)+ 1-oo; x6end =  (file_count*6)-oo;
x6 = v(x6start : x6end, modes);
x7start = (6*file_count)+ 1-oo; x7end = (file_count*7)-oo;
x7 = v(x7start : x7end, modes);
x8start = (7*file_count)+ 1-oo; x8end = (file_count*8)-oo;
```

```
x8 = v(x8start : x8end, modes);
% setup train/test datasets
ntr = 76;
tr = 152;
xtrain = [x1(q1(1:ntr), :); x2(q2(1:tr), :); x3(q3(1:tr), :); x4(q4(1:tr), :); x5(q5(1:tr), :); x6(q6(1:tr), :); x7(q7(1:tr), :);
x8(q8(1:tr), :)];
xtest = [x1(q1(ntr+1:end), :); x2(q2(tr+1:end), :); x3(q3(tr+1:end), :); x4(q4(tr+1:end), :); x5(q5(tr+1:end), :);
x6(q6(tr+1:end), :); x7(q7(tr+1:end), :); x8(q8(tr+1:end), :)];
ctrain = [ones(ntr, 1); 2*ones(tr, 1); 3*ones(tr, 1); 4*ones(tr, 1); 5*ones(tr, 1); 6*ones(tr, 1); 7*ones(tr, 1); 8*ones(tr, 1)];
% linear discrimination analysis
pre1 = classify(xtest, xtrain, ctrain);
exp_class = [ones(20, 1); 2*ones(40, 1); 3*ones(40, 1); 4*ones(40, 1); 5*ones(40, 1); 6*ones(40, 1); 7*ones(40, 1);
8*ones(40, 1)];
error1 = nnz(pre1(1:20) - 1);
accur1 = (20-error1)/20;
error2 = nnz(pre1(21:60) - 2);
accur2 = (40-error2)/40;
error3 = nnz(pre1(61:100) - 3);
accur3 = (40-error3)/40 ;
error4 = nnz(pre1(101:140) - 4);
accur4 = (40-error4)/40 ;
error5 = nnz(pre1(141:180) - 5);
accur5 = (40-error5)/40 ;
error6 = nnz(pre1(181:220) - 6);
accur6 = (40-error6)/40 ;
error7 = nnz(pre1(221:260) - 7);
accur7 = (40-error7)/40 ;
error8 = nnz(pre1(261:300) - 8);
accur8 = (40-error8)/40 ;
ld_accur2(jj, :)  = horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
ld_mean_accuracy2 = mean(ld_accur2, 1);
% naive bayes
nb = fitcnb(xtrain, ctrain);
pre2 = nb.predict(xtest);
error1 = nnz(pre2(1:20) - 1);
accur1 = (20-error1)/20;
error2 = nnz(pre2(21:60) - 2);
accur2 = (40-error2)/40;
error3 = nnz(pre2(61:100) - 3);
accur3 = (40-error3)/40 ;
error4 = nnz(pre2(101:140) - 4);
accur4 = (40-error4)/40 ;
error5 = nnz(pre2(141:180) - 5);
accur5 = (40-error5)/40 ;
error6 = nnz(pre2(181:220) - 6);
accur6 = (40-error6)/40 ;
error7 = nnz(pre2(221:260) - 7);
accur7 = (40-error7)/40 ;
error8 = nnz(pre2(261:300) - 8);
accur8 = (40-error8)/40 ;
nb_accur2(jj, :)  =  horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
nb_mean_accuracy2 = mean(nb_accur2, 1);

% k nearest neighbor
```

```
[ind, D] = knnsearch(xtrain, xtest);

for ii = 1:length(ind)
    pre3(ii, :) = ctrain(ind(ii));
end

error1 = nnz(pre3(1:20) - 1);
accur1 = (20-error1)/20;
error2 = nnz(pre3(21:60) - 2);
accur2 = (40-error2)/40;
error3 = nnz(pre3(61:100) - 3);
accur3 = (40-error3)/40 ;
error4 = nnz(pre3(101:140) - 4);
accur4 = (40-error4)/40 ;
error5 = nnz(pre3(141:180) - 5);
accur5 = (40-error5)/40 ;
error6 = nnz(pre3(181:220) - 6);
accur6 = (40-error6)/40 ;
error7 = nnz(pre3(221:260) - 7);
accur7 = (40-error7)/40 ;
error8 = nnz(pre3(261:300) - 8);
accur8 = (40-error8)/40 ;

knn_accur2(jj, :)  = horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
knn_mean_accuracy2 = mean(knn_accur2, 1);
end
results2(mm, :) = [ld_mean_accuracy2, nb_mean_accuracy2,knn_mean_accuracy2];
end
%% highest performing modes for LD and NB Classification Methods
neutral = results2(:, [1 9 17]);
calm = results2(:, [2 10 18]);
happy = results2(:, [3 11 19]);
sad = results2(:, [4 12 20]);
angry = results2(:, [5 13 21]);
fearful = results2(:, [6 14 22]);
disgust = results2(:, [7 15 23]);
surprised = results2(:, [8 16 24]);

[M1, I1] = max(neutral(:, 1)); [n_mode, n_class] = ind2sub(size(neutral(:, 1)), I1);
[M2, I2] = max(calm(:, 1)); [c_mode, c_class] = ind2sub(size(calm(:, 1)), I2);
[M3, I3] = max(happy(:, 1)); [h_mode, h_class] = ind2sub(size(happy(:, 1)), I3);
[M4, I4] = max(sad(:, 1)); [s_mode, s_class] = ind2sub(size(sad(:, 1)), I4);
[M5, I5] = max(angry(:, 1)); [a_mode, a_class] = ind2sub(size(angry(:, 1)), I5);
[M6, I6] = max(fearful(:, 1));[f_mode, f_class] = ind2sub(size(fearful(:, 1)), I6);
[M7, I7] = max(disgust(:, 1));[d_mode, d_class] = ind2sub(size(disgust(:, 1)), I7);
[M8, I8] = max(surprised(:, 1));[su_mode, su_class] = ind2sub(size(surprised(:, 1)), I8);
cm_ld = [n_mode M1; c_mode M2; h_mode M3; s_mode M4; a_mode M5;  f_mode M6; d_mode M7; su_mode M8];
cmtable_ld = table( cm_ld(:, 1),cm_ld(:, 2), 'VariableNames', {'Mode'; 'Accuracy'});
%
[M1, I1] = max(neutral(:, 2)); [n_mode, n_class] = ind2sub(size(neutral(:, 2)), I1);
[M2, I2] = max(calm(:, 2)); [c_mode, c_class] = ind2sub(size(calm(:, 2)), I2);
[M3, I3] = max(happy(:, 2)); [h_mode, h_class] = ind2sub(size(happy(:, 2)), I3);
[M4, I4] = max(sad(:, 2)); [s_mode, s_class] = ind2sub(size(sad(:, 2)), I4);
[M5, I5] = max(angry(:, 2)); [a_mode, a_class] = ind2sub(size(angry(:, 2)), I5);
```

```
[M6, I6] = max(fearful(:, 2));[f_mode, f_class] = ind2sub(size(fearful(:, 2)), I6);
[M7, I7] = max(disgust(:, 2));[d_mode, d_class] = ind2sub(size(disgust(:, 2)), I7);
[M8, I8] = max(surprised(:, 2));[su_mode, su_class] = ind2sub(size(surprised(:, 2)), I8);
cm_nb = [n_mode M1; c_mode M2; h_mode M3; s_mode M4; a_mode M5;  f_mode M6; d_mode M7; su_mode M8];
cmtable_nb = table(cm_nb(:, 1),cm_nb(:,2), 'VariableNames', {'Mode'; 'Accuracy'});
%
[M1, I1] = max(neutral(:, 3)); [n_mode, n_class] = ind2sub(size(neutral(:, 3)), I1);
[M2, I2] = max(calm(:, 3)); [c_mode, c_class] = ind2sub(size(calm(:, 3)), I2);
[M3, I3] = max(happy(:, 3)); [h_mode, h_class] = ind2sub(size(happy(:, 3)), I3);
[M4, I4] = max(sad(:, 3)); [s_mode, s_class] = ind2sub(size(sad(:, 3)), I4);
[M5, I5] = max(angry(:, 3)); [a_mode, a_class] = ind2sub(size(angry(:, 3)), I5);
[M6, I6] = max(fearful(:, 3));[f_mode, f_class] = ind2sub(size(fearful(:, 3)), I6);
[M7, I7] = max(disgust(:, 3));[d_mode, d_class] = ind2sub(size(disgust(:, 3)), I7);
[M8, I8] = max(surprised(:, 3));[su_mode, su_class] = ind2sub(size(surprised(:, 3)), I8);
cm_knn = [n_mode M1; c_mode M2; h_mode M3; s_mode M4; a_mode M5;  f_mode M6; d_mode M7; su_mode
M8];
cmtable_knn = table(cm_knn(:, 1),cm_knn(:,2), 'VariableNames', {'Mode'; 'Accuracy'});

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Based on single mode/class results, mode select and classification of NB%%%

modes = horzcat(cm_ld(:, 1),cm_nb(:, 1), cm_knn(:, 1))';
%%
for ss = 1:3
for jj = 1:50
q1 = randperm(96);
q2 = randperm(file_count);
q3 = randperm(file_count);
q4 = randperm(file_count);
q5 = randperm(file_count);
q6 = randperm(file_count);
q7 = randperm(file_count);
q8 = randperm(file_count);
oo = 96;% neutral
x1start = 1; x1end = file_count-oo;
x1= v(x1start : x1end, modes(ss, :));
x2start = file_count+1-oo; x2end =(file_count*2)-oo;
x2 = v(x2start : x2end,modes(ss, :));
x3start = (2*file_count)+ 1-oo; x3end = (file_count*3)-oo;
x3 = v(x3start : x3end, modes(ss, :));
x4start = (3*file_count)+ 1-oo;  x4end =(file_count*4)-oo;
x4 = v(x4start : x4end, modes(ss, :));
x5start = (4*file_count)+ 1-oo;  x5end =(file_count*5)-oo;
x5 = v(x5start : x5end, modes(ss, :));
x6start = (5*file_count)+ 1-oo; x6end =  (file_count*6)-oo;
x6 = v(x6start : x6end, modes(ss, :));
x7start = (6*file_count)+ 1-oo; x7end = (file_count*7)-oo;
x7 = v(x7start : x7end, modes(ss, :));
x8start = (7*file_count)+ 1-oo; x8end = (file_count*8)-oo;
x8 = v(x8start : x8end, modes(ss, :));
% setup train/test datasets
ntr = 76;
tr = 152;
```

```
xtrain = [x1(q1(1:ntr), :); x2(q2(1:tr), :); x3(q3(1:tr), :); x4(q4(1:tr), :); x5(q5(1:tr), :); x6(q6(1:tr), :); x7(q7(1:tr), :);
x8(q8(1:tr), :)];
xtest = [x1(q1(ntr+1:end), :); x2(q2(tr+1:end), :); x3(q3(tr+1:end), :); x4(q4(tr+1:end), :); x5(q5(tr+1:end), :);
x6(q6(tr+1:end), :); x7(q7(tr+1:end), :); x8(q8(tr+1:end), :)];
ctrain = [ones(ntr, 1); 2*ones(tr, 1); 3*ones(tr, 1); 4*ones(tr, 1); 5*ones(tr, 1); 6*ones(tr, 1); 7*ones(tr, 1); 8*ones(tr, 1)];

exp_class = [ones(20, 1); 2*ones(40, 1); 3*ones(40, 1); 4*ones(40, 1); 5*ones(40, 1); 6*ones(40, 1); 7*ones(40, 1);
8*ones(40, 1)];
% linear discrimination
if ss ==1
 pre1 = classify(xtest, xtrain, ctrain);
exp_class = [ones(20, 1); 2*ones(40, 1); 3*ones(40, 1); 4*ones(40, 1); 5*ones(40, 1); 6*ones(40, 1); 7*ones(40, 1);
8*ones(40, 1)];
error1 = nnz(pre1(1:20) - 1);
accur1 = (20-error1)/20;
error2 = nnz(pre1(21:60) - 2);
accur2 = (40-error2)/40;
error3 = nnz(pre1(61:100) - 3);
accur3 = (40-error3)/40 ;
error4 = nnz(pre1(101:140) - 4);
accur4 = (40-error4)/40 ;
error5 = nnz(pre1(141:180) - 5);
accur5 = (40-error5)/40 ;
error6 = nnz(pre1(181:220) - 6);
accur6 = (40-error6)/40 ;
error7 = nnz(pre1(221:260) - 7);
accur7 = (40-error7)/40 ;
error8 = nnz(pre1(261:300) - 8);
accur8 = (40-error8)/40 ;

ld_accur3(jj, :)  = horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
ld_mean_accuracy3 = mean(ld_accur3, 1);
else
end
% naive bayes
if ss ==2
nb = fitcnb(xtrain, ctrain);
pre2 = nb.predict(xtest);

error1 = nnz(pre2(1:20) - 1);
accur1 = (20-error1)/20;
error2 = nnz(pre2(21:60) - 2);
accur2 = (40-error2)/40;
error3 = nnz(pre2(61:100) - 3);
accur3 = (40-error3)/40 ;
error4 = nnz(pre2(101:140) - 4);
accur4 = (40-error4)/40 ;
error5 = nnz(pre2(141:180) - 5);
accur5 = (40-error5)/40 ;
error6 = nnz(pre2(181:220) - 6);
accur6 = (40-error6)/40 ;
error7 = nnz(pre2(221:260) - 7);
accur7 = (40-error7)/40 ;
error8 = nnz(pre2(261:300) - 8);
```

```
accur8 = (40-error8)/40 ;

nb_accur3(jj, :)  =  horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
nb_mean_accuracy3 = mean(nb_accur3, 1);
else
end
% k nearest neighbor
if ss == 3
[ind, D] = knnsearch(xtrain, xtest);

for ii = 1:length(ind)
    pre3(ii, :) = ctrain(ind(ii));
end

error1 = nnz(pre3(1:20) - 1);
accur1 = (20-error1)/20;
error2 = nnz(pre3(21:60) - 2);
accur2 = (40-error2)/40;
error3 = nnz(pre3(61:100) - 3);
accur3 = (40-error3)/40 ;
error4 = nnz(pre3(101:140) - 4);
accur4 = (40-error4)/40 ;
error5 = nnz(pre3(141:180) - 5);
accur5 = (40-error5)/40 ;
error6 = nnz(pre3(181:220) - 6);
accur6 = (40-error6)/40 ;
error7 = nnz(pre3(221:260) - 7);
accur7 = (40-error7)/40 ;
error8 = nnz(pre3(261:300) - 8);
accur8 = (40-error8)/40 ;

knn_accur3(jj, :)  = horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
knn_mean_accuracy3 = mean(knn_accur3, 1);
else
end
end
end

figure(4)
sgtitle('Accuracy of Classification Methods using "high performing"  Modes');
plot(x, ld_mean_accuracy3*100, 'b-*'); hold on
plot(x, nb_mean_accuracy3*100, 'r-*');
plot(x, knn_mean_accuracy3*100, 'k-*');
legend( 'Linear Discrimination', 'Naive Bayes', 'knn', 'Location', 'NorthEast')
xlim([.8 8.1]); xlabel('Classification')
ylim([0 102])
ylabel('Accuracy (%)')
```

Script 2: (emotion_male.m)

```
% load male files, truncate, fourier transform
% Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
clear all; close all; clc;
n = 140900;
% emotion = neutral
```

```
pgmfiles=dir('01_male/*/03-01-01-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y1, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y1 = y1(1:n);
   y1_all(:, ii) = y1;
   v1 = y1'; y1f = fft(v1); y1fs = abs(fftshift(y1f));
   y1fs_all(:, ii) = y1fs;
end
% emotion = calm
pgmfiles=dir('01_male/*/03-01-02-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y2, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y2 = y2(1:n);
   y2_all(:, ii) = y2;
   v2 = y2'; y2f = fft(v2); y2fs = abs(fftshift(y2f));
   y2fs_all(:, ii) = y2fs;
end
% emotion = happy
pgmfiles=dir('01_male/*/03-01-03-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y3, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y3 = y3(1:n);
   y3_all(:, ii) = y3;
   v3 = y3'; y3f = fft(v3); y3fs = abs(fftshift(y3f));
   y3fs_all(:, ii) = y3fs;
end
% emotion = sad
pgmfiles=dir('01_male/*/03-01-04-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y4, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y4 = y4(1:n);
   y4_all(:, ii) = y4;
   v4 = y4'; y4f = fft(v4); y4fs = abs(fftshift(y4f));
   y4fs_all(:, ii) = y4fs;
end
% emotion = angry
pgmfiles=dir('01_male/*/03-01-05-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y5, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y5= y5(1:n);
   y5_all(:, ii) = y5;
   v5 = y5'; y5f = fft(v5); y5fs = abs(fftshift(y5f));
   y5fs_all(:, ii) = y5fs;
end
% emotion = fearful
pgmfiles=dir('01_male/*/03-01-06-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y6, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y6= y6(1:n);
   y6_all(:, ii) = y6;
   v6 = y6'; y6f = fft(v6); y6fs = abs(fftshift(y6f));
   y6fs_all(:, ii) = y6fs;
end
% emotion = disgust
```

```
pgmfiles=dir('01_male/*/03-01-07-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y7, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y7= y7(1:n);
    y7_all(:, ii) = y7;
    v7 = y7'; y7f = fft(v7); y7fs = abs(fftshift(y7f));
    y7fs_all(:, ii) = y7fs;
end
% emotion = surprise
pgmfiles=dir('01_male/*/03-01-08-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y8, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y8= y8(1:n);
    y8_all(:, ii) = y8;
    v8 = y8'; y8f = fft(v8); y8fs = abs(fftshift(y8f));
    y8fs_all(:, ii) = y8fs;
end
y_all = horzcat(y1_all, y2_all, y3_all, y4_all, y5_all, y6_all, y7_all, y8_all);
yfs_all = horzcat(y1fs_all, y2fs_all, y3fs_all, y4fs_all, y5fs_all, y6fs_all, y7fs_all, y8fs_all);

%  normalize (take mean along rows) and perform SVD to find singular values
[m,n] = size(yfs_all);
mn=mean(yfs_all,2); %compute mean for each row
clips_mean=yfs_all -repmat(mn, 1, n); % subtract mean
[u,s,v] = svd(clips_mean', 'econ');
 sig = diag(s);
energy_all = zeros(n, 2);

%calculate cumulative energy
for ee = 1:n
    energy_s = sum(sig(1:ee))/sum(sig) ;
    energy_all(ee, 2) = energy_s;
    energy_all(ee, 1) = ee;
end
%%
figure(1)
 sgtitle('Audio Clips (MALE):  Singular Values (\sigma) by mode')
 subplot(2,2,1)
plot(sig, 'bo', 'Linewidth', [1]);
xlabel('mode')
ylabel('\sigma')
subplot(2,2,2)
semilogy(sig,'bo','Linewidth',[1])
xlabel('mode')
ylabel('\sigma (log)')
subplot(2,1, 2)
plot(energy_all(:, 2)*100, 'b:', 'Linewidth', [4])
title('Cumulative Energy by mode')
ylabel('% energy (\sigma)')
xlabel('Cumulative Modes')
%%
% calculate table with cum energy at 10% segment  thresholds
energy_summ = zeros(10, 2);
for ee = 0.1:0.1: 1
```

```
thres = find(energy_all(:,2) >= ee);
min_loc = uint64(min(thres));
energy_summ(uint64(ee*10), :) = energy_all(min_loc, :);
end
s_table = table(energy_summ(:, 1), energy_summ(:, 2), 'VariableNames', {'Mode'; 'Energy %'});

%% Supervised Learning
% Cross valudation using 100 iter of sampled train vs test data.
m_vector = [30:10:90];
for mm = m_vector
modes = min(find(energy_all(:, 2) >mm/100));
for jj = 1:50
q1 = randperm(48);
q2 = randperm(file_count);
q3 = randperm(file_count);
q4 = randperm(file_count);
q5 = randperm(file_count);
q6 = randperm(file_count);
q7 = randperm(file_count);
q8 = randperm(file_count);
oo = 48;% neutral
x1start = 1; x1end = file_count-oo;
x1= v(x1start : x1end, 1:modes);
x2start = file_count+1-oo; x2end =(file_count*2)-oo;
x2 = v(x2start : x2end, 1:modes);
x3start = (2*file_count)+ 1-oo; x3end = (file_count*3)-oo;
x3 = v(x3start : x3end, 1:modes);
x4start = (3*file_count)+ 1-oo;  x4end =(file_count*4)-oo;
x4 = v(x4start : x4end, 1:modes);
x5start = (4*file_count)+ 1-oo;  x5end =(file_count*5)-oo;
x5 = v(x5start : x5end, 1:modes);
x6start = (5*file_count)+ 1-oo; x6end =  (file_count*6)-oo;
x6 = v(x6start : x6end, 1:modes);
x7start = (6*file_count)+ 1-oo; x7end = (file_count*7)-oo;
x7 = v(x7start : x7end, 1:modes);
x8start = (7*file_count)+ 1-oo; x8end = (file_count*8)-oo;
x8 = v(x8start : x8end, 1:modes);
% setup train/test datasets
ntr = 38;
tr = 76;
ntst = 10;
tst = 20;
xtrain = [x1(q1(1:ntr), :); x2(q2(1:tr), :); x3(q3(1:tr), :); x4(q4(1:tr), :); x5(q5(1:tr), :); x6(q6(1:tr), :); x7(q7(1:tr), :);
x8(q8(1:tr), :)];
xtest = [x1(q1(ntr+1:end), :); x2(q2(tr+1:end), :); x3(q3(tr+1:end), :); x4(q4(tr+1:end), :); x5(q5(tr+1:end), :);
x6(q6(tr+1:end), :); x7(q7(tr+1:end), :); x8(q8(tr+1:end), :)];
ctrain = [ones(ntr, 1); 2*ones(tr, 1); 3*ones(tr, 1); 4*ones(tr, 1); 5*ones(tr, 1); 6*ones(tr, 1); 7*ones(tr, 1); 8*ones(tr, 1)];

% linear discrimination analysis
pre1 = classify(xtest, xtrain, ctrain);
exp_class = [ones(ntst, 1); 2*ones(tst, 1); 3*ones(tst, 1); 4*ones(tst, 1); 5*ones(tst, 1); 6*ones(tst, 1); 7*ones(tst, 1);
8*ones(tst, 1)];
error1 = nnz(pre1(1:10) - 1);
accur1 = (10-error1)/10;
```

```
error2 = nnz(pre1(11:30) - 2);
accur2 = (20-error2)/20;
error3 = nnz(pre1(31:50) - 3);
accur3 = (20-error3)/20 ;
error4 = nnz(pre1(51:70) - 4);
accur4 = (20-error4)/20 ;
error5 = nnz(pre1(71:90) - 5);
accur5 = (20-error5)/20 ;
error6 = nnz(pre1(91:110) - 6);
accur6 = (20-error6)/20 ;
error7 = nnz(pre1(111:130) - 7);
accur7 = (20-error7)/20 ;
error8 = nnz(pre1(131:150) - 8);
accur8 = (20-error8)/20 ;

ld_accur(jj, :)  = horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
ld_mean_accuracy = mean(ld_accur, 1);
% naive bayes
nb = fitcnb(xtrain, ctrain);
pre2 = nb.predict(xtest);

error1 = nnz(pre2(1:10) - 1);
accur1 = (10-error1)/10;
error2 = nnz(pre2(11:30) - 2);
accur2 = (20-error2)/20;
error3 = nnz(pre2(31:50) - 3);
accur3 = (20-error3)/20 ;
error4 = nnz(pre2(51:70) - 4);
accur4 = (20-error4)/20 ;
error5 = nnz(pre2(71:90) - 5);
accur5 = (20-error5)/20 ;
error6 = nnz(pre2(91:110) - 6);
accur6 = (20-error6)/20 ;
error7 = nnz(pre2(111:130) - 7);
accur7 = (20-error7)/20 ;
error8 = nnz(pre2(131:150) - 8);
accur8 = (20-error8)/20 ;

nb_accur(jj, :)  =  horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
nb_mean_accuracy = mean(nb_accur, 1);

% k nearest neighbor
[ind, D] = knnsearch(xtrain, xtest);

for ii = 1:length(ind)
    pre3(ii, :) = ctrain(ind(ii));
end

error1 = nnz(pre3(1:10) - 1);
accur1 = (10-error1)/10;
error2 = nnz(pre3(11:30) - 2);
accur2 = (20-error2)/20;
error3 = nnz(pre3(31:50) - 3);
accur3 = (20-error3)/20 ;
```

```
error4 = nnz(pre3(51:70) - 4);
accur4 = (20-error4)/20 ;
error5 = nnz(pre3(71:90) - 5);
accur5 = (20-error5)/20 ;
error6 = nnz(pre3(91:110) - 6);
accur6 = (20-error6)/20 ;
error7 = nnz(pre3(111:130) - 7);
accur7 = (20-error7)/20 ;
error8 = nnz(pre3(131:150) - 8);
accur8 = (20-error8)/20 ;

knn_accur(jj, :)  = horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
knn_mean_accuracy = mean(knn_accur, 1);

end
results((mm/10)-2, :) = [ld_mean_accuracy, nb_mean_accuracy,knn_mean_accuracy];
end

figure (2)
sgtitle('Emotion Classification (MALE): Sample Iteration')
subplot(1, 3, 1);
bar(pre1); hold on;
title('Linear Discrimination')
plot(exp_class, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip No.')
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;
subplot(1, 3, 2);
bar(pre2); hold on;
title('Naive Bayes')
plot(exp_class, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip No.')
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;

subplot(1, 3, 3);

bar(pre3); hold on;
title('K Nearest Neighbor')
plot(exp_class, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip No.')
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;

%%
figure(3)
sgtitle('Emotion Classification (MALE) : Cumulative Energy Comparison')
x = 1:8;
subplot(1,3, 1)
plot(x, results(1, 1:8)*100, 'b-o'); hold on;
plot(x, results(2, 1:8)*100, 'r-o');
plot(x, results(3, 1:8)*100, 'g-o');
plot(x, results(4, 1:8)*100, 'm-o');
plot(x, results(5, 1:8)*100, 'c-o');
plot(x, results(6, 1:8)*100, '-o','Color',[.8,.4,.2]);
plot(x, results(7, 1:8)*100, 'k-o');
xlim([.8 8.1]); xlabel('Classification')
ylim([0 102])
title('Linear Discrimination')
legend( '30%','40%', '50%', '60%','70%', '80%','90%', 'Location', 'NorthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3,2)
plot(x, results(1,9:16)*100, 'b-o'); hold on;
plot(x, results(2, 9:16)*100, 'r-o');
plot(x, results(3, 9:16)*100, 'g-o');
plot(x, results(4, 9:16)*100, 'm-o');
plot(x, results(5, 9:16)*100, 'c-o');
plot(x, results(6, 9:16)*100,'-o','Color',[.8,.4,.2]);
plot(x, results(7, 9:16)*100, 'k-o');
xlim([.8 8.1]); xlabel('Classification')
ylim([0 102])
title('Naive Bayes')
legend( '30%','40%', '50%', '60%','70%', '80%','90%', 'Location', 'NorthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3, 3)
plot(x, results(1, 17:24)*100, 'b-o'); hold on;
plot(x, results(2, 17:24)*100, 'r-o');
plot(x, results(3,17:24)*100, 'g-o');
plot(x, results(4, 17:24)*100, 'm-o');
plot(x, results(5, 17:24)*100, 'c-o');
plot(x, results(6, 17:24)*100, '-o','Color',[.8,.4,.2]);
plot(x, results(7, 17:24)*100, 'k-o');
xlim([.8 8.1]); xlabel('Classification')
ylim([0 102])
title('K Nearest Neighbor')
legend( '30%','40%', '50%', '60%','70%', '80%','90%', 'Location', 'NorthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
```

Script 3: (emotion_female.m)

```
% load female files, truncate,  fourier transform
% Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
clear all; close all; clc;
n = 140900;
% emotion = neutral
pgmfiles=dir('02_female/*/03-01-01-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
   [y1, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y1 = y1(1:n);
   y1_all(:, ii) = y1;
   v1 = y1'; y1f = fft(v1); y1fs = abs(fftshift(y1f));
```

```
    y1fs_all(:, ii) = y1fs;
end
% emotion = calm
pgmfiles=dir('02_female/*/03-01-02-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y2, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y2 = y2(1:n);
    y2_all(:, ii) = y2;
    v2 = y2'; y2f = fft(v2); y2fs = abs(fftshift(y2f));
    y2fs_all(:, ii) = y2fs;
end
% emotion = happy
pgmfiles=dir('02_female/*/03-01-03-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y3, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y3 = y3(1:n);
    y3_all(:, ii) = y3;
    v3 = y3'; y3f = fft(v3); y3fs = abs(fftshift(y3f));
    y3fs_all(:, ii) = y3fs;
end
% emotion = sad
pgmfiles=dir('02_female/*/03-01-04-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y4, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y4 = y4(1:n);
    y4_all(:, ii) = y4;
    v4 = y4'; y4f = fft(v4); y4fs = abs(fftshift(y4f));
    y4fs_all(:, ii) = y4fs;
end
% emotion = angry
pgmfiles=dir('02_female/*/03-01-05-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y5, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y5= y5(1:n);
    y5_all(:, ii) = y5;
    v5 = y5'; y5f = fft(v5); y5fs = abs(fftshift(y5f));
    y5fs_all(:, ii) = y5fs;
end
% emotion = fearful
pgmfiles=dir('02_female/*/03-01-06-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y6, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y6= y6(1:n);
    y6_all(:, ii) = y6;
    v6 = y6'; y6f = fft(v6); y6fs = abs(fftshift(y6f));
    y6fs_all(:, ii) = y6fs;
end
% emotion = disgust
pgmfiles=dir('02_female/*/03-01-07-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y7, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y7= y7(1:n);
    y7_all(:, ii) = y7;
    v7 = y7'; y7f = fft(v7); y7fs = abs(fftshift(y7f));
    y7fs_all(:, ii) = y7fs;
end
% emotion = surprise
pgmfiles=dir('02_female/*/03-01-08-*.wav'); file_count = length(pgmfiles); % load filesnames and count
for ii = 1:file_count
    [y8, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y8= y8(1:n);
    y8_all(:, ii) = y8;
    v8 = y8'; y8f = fft(v8); y8fs = abs(fftshift(y8f));
    y8fs_all(:, ii) = y8fs;
end
y_all = horzcat(y1_all, y2_all, y3_all, y4_all, y5_all, y6_all, y7_all, y8_all);
yfs_all = horzcat(y1fs_all, y2fs_all, y3fs_all, y4fs_all, y5fs_all, y6fs_all, y7fs_all, y8fs_all);

%  normalize (take mean along rows) and perform SVD to find singular values
[m,n] = size(yfs_all);
mn=mean(yfs_all,2); %compute mean for each row
clips_mean=yfs_all -repmat(mn, 1, n); % subtract mean
[u,s,v] = svd(clips_mean', 'econ');
 sig = diag(s);
energy_all = zeros(n, 2);

%calculate cumulative energy
for ee = 1:n
    energy_s = sum(sig(1:ee))/sum(sig) ;
    energy_all(ee, 2) = energy_s;
    energy_all(ee, 1) = ee;
end
%%
figure(1)
 sgtitle('Audio Clips (FEMALE):  Singular Values (\sigma) by mode')
 subplot(2,2,1)
plot(sig, 'bo', 'Linewidth', [1]);
xlabel('mode')
ylabel('\sigma')
subplot(2,2,2)
semilogy(sig,'bo','Linewidth',[1])
xlabel('mode')
ylabel('\sigma (log)')
subplot(2,1, 2)
plot(energy_all(:, 2)*100, 'b:', 'Linewidth', [4])
title('Cumulative Energy by mode')
ylabel('% energy (\sigma)')
%%
% calculate table with cum energy at 10% segment  thresholds
energy_summ = zeros(10, 2);
for ee = 0.1:0.1: 1
thres = find(energy_all(:,2) >= ee);
min_loc = uint64(min(thres));
energy_summ(uint64(ee*10), :) = energy_all(min_loc, :);
end
s_table = table(energy_summ(:, 1), energy_summ(:, 2), 'VariableNames', {'Mode'; 'Energy %'});

%% Supervised Learning
```

```
% Cross valudation using 100 iter of sampled train vs test data.
m_vector = [30:10:90];
for mm = m_vector
modes = min(find(energy_all(:, 2) >mm/100));
for jj = 1:50
q1 = randperm(48);
q2 = randperm(file_count);
q3 = randperm(file_count);
q4 = randperm(file_count);
q5 = randperm(file_count);
q6 = randperm(file_count);
q7 = randperm(file_count);
q8 = randperm(file_count);
oo = 48;% neutral
x1start = 1; x1end = file_count-oo;
x1= v(x1start : x1end, 1:modes);
x2start = file_count+1-oo; x2end =(file_count*2)-oo;
x2 = v(x2start : x2end, 1:modes);
x3start = (2*file_count)+ 1-oo; x3end = (file_count*3)-oo;
x3 = v(x3start : x3end, 1:modes);
x4start = (3*file_count)+ 1-oo;  x4end =(file_count*4)-oo;
x4 = v(x4start : x4end, 1:modes);
x5start = (4*file_count)+ 1-oo;  x5end =(file_count*5)-oo;
x5 = v(x5start : x5end, 1:modes);
x6start = (5*file_count)+ 1-oo; x6end =  (file_count*6)-oo;
x6 = v(x6start : x6end, 1:modes);
x7start = (6*file_count)+ 1-oo; x7end = (file_count*7)-oo;
x7 = v(x7start : x7end, 1:modes);
x8start = (7*file_count)+ 1-oo; x8end = (file_count*8)-oo;
x8 = v(x8start : x8end, 1:modes);
% setup train/test datasets
ntr = 38;
tr = 76;
ntst = 10;
tst = 20;
xtrain = [x1(q1(1:ntr), :); x2(q2(1:tr), :); x3(q3(1:tr), :); x4(q4(1:tr), :); x5(q5(1:tr), :); x6(q6(1:tr), :); x7(q7(1:tr), :);
x8(q8(1:tr), :)];
xtest = [x1(q1(ntr+1:end), :); x2(q2(tr+1:end), :); x3(q3(tr+1:end), :); x4(q4(tr+1:end), :); x5(q5(tr+1:end), :);
x6(q6(tr+1:end), :); x7(q7(tr+1:end), :); x8(q8(tr+1:end), :)];
ctrain = [ones(ntr, 1); 2*ones(tr, 1); 3*ones(tr, 1); 4*ones(tr, 1); 5*ones(tr, 1); 6*ones(tr, 1); 7*ones(tr, 1); 8*ones(tr, 1)];

% linear discrimination analysis
pre1 = classify(xtest, xtrain, ctrain);
exp_class = [ones(ntst, 1); 2*ones(tst, 1); 3*ones(tst, 1); 4*ones(tst, 1); 5*ones(tst, 1); 6*ones(tst, 1); 7*ones(tst, 1);
8*ones(tst, 1)];
error1 = nnz(pre1(1:10) - 1);
accur1 = (10-error1)/10;
error2 = nnz(pre1(11:30) - 2);
accur2 = (20-error2)/20;
error3 = nnz(pre1(31:50) - 3);
accur3 = (20-error3)/20 ;
error4 = nnz(pre1(51:70) - 4);
accur4 = (20-error4)/20 ;
error5 = nnz(pre1(71:90) - 5);
```

```
accur5 = (20-error5)/20 ;
error6 = nnz(pre1(91:110) - 6);
accur6 = (20-error6)/20 ;
error7 = nnz(pre1(111:130) - 7);
accur7 = (20-error7)/20 ;
error8 = nnz(pre1(131:150) - 8);
accur8 = (20-error8)/20 ;

ld_accur(jj, :)  = horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
ld_mean_accuracy = mean(ld_accur, 1);
% naive bayes
nb = fitcnb(xtrain, ctrain);
pre2 = nb.predict(xtest);

error1 = nnz(pre2(1:10) - 1);
accur1 = (10-error1)/10;
error2 = nnz(pre2(11:30) - 2);
accur2 = (20-error2)/20;
error3 = nnz(pre2(31:50) - 3);
accur3 = (20-error3)/20 ;
error4 = nnz(pre2(51:70) - 4);
accur4 = (20-error4)/20 ;
error5 = nnz(pre2(71:90) - 5);
accur5 = (20-error5)/20 ;
error6 = nnz(pre2(91:110) - 6);
accur6 = (20-error6)/20 ;
error7 = nnz(pre2(111:130) - 7);
accur7 = (20-error7)/20 ;
error8 = nnz(pre2(131:150) - 8);
accur8 = (20-error8)/20 ;

nb_accur(jj, :)  =  horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
nb_mean_accuracy = mean(nb_accur, 1);

% k nearest neighbor
[ind, D] = knnsearch(xtrain, xtest);

for ii = 1:length(ind)
   pre3(ii, :) = ctrain(ind(ii));
end

error1 = nnz(pre3(1:10) - 1);
accur1 = (10-error1)/10;
error2 = nnz(pre3(11:30) - 2);
accur2 = (20-error2)/20;
error3 = nnz(pre3(31:50) - 3);
accur3 = (20-error3)/20 ;
error4 = nnz(pre3(51:70) - 4);
accur4 = (20-error4)/20 ;
error5 = nnz(pre3(71:90) - 5);
accur5 = (20-error5)/20 ;
error6 = nnz(pre3(91:110) - 6);
accur6 = (20-error6)/20 ;
error7 = nnz(pre3(111:130) - 7);
```

```
accur7 = (20-error7)/20 ;
error8 = nnz(pre3(131:150) - 8);
accur8 = (20-error8)/20 ;

knn_accur(jj, :)  = horzcat(accur1, accur2, accur3, accur4, accur5, accur6, accur7, accur8);
knn_mean_accuracy = mean(knn_accur, 1);

end
results((mm/10)-2, :) = [ld_mean_accuracy, nb_mean_accuracy,knn_mean_accuracy];
end

figure (2)
sgtitle('Emotion Classification (FEMALE): Sample Iteration')
subplot(1, 3, 1);
bar(pre1); hold on;
title('Linear Discrimination')
plot(exp_class, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip No.')
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;
subplot(1, 3, 2);
bar(pre2); hold on;
title('Naive Bayes')
plot(exp_class, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip No.')
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;

subplot(1, 3, 3);

bar(pre3); hold on;
title('K Nearest Neighbor')
plot(exp_class, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip No.')
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;

%%
figure(3)
sgtitle('Emotion Classification (FEMALE) : Cumulative Energy Comparison')
x = 1:8;
subplot(1,3, 1)
plot(x, results(1, 1:8)*100, 'b-o'); hold on;
plot(x, results(2, 1:8)*100, 'r-o');
plot(x, results(3, 1:8)*100, 'g-o');
plot(x, results(4, 1:8)*100, 'm-o');
plot(x, results(5, 1:8)*100, 'c-o');
plot(x, results(6, 1:8)*100, '-o','Color',[.8,.4,.2]);
plot(x, results(7, 1:8)*100, 'k-o');
xlim([.8 8.1]); xlabel('Classification')
ylim([0 102])
```

```
title('Linear Discrimination')
legend( '30%','40%', '50%', '60%','70%', '80%','90%', 'Location', 'NorthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3,2)
plot(x, results(1,9:16)*100, 'b-o'); hold on;
plot(x, results(2, 9:16)*100, 'r-o');
plot(x, results(3, 9:16)*100, 'g-o');
plot(x, results(4, 9:16)*100, 'm-o');
plot(x, results(5, 9:16)*100, 'c-o');
plot(x, results(6, 9:16)*100,'-o','Color',[.8,.4,.2]);
plot(x, results(7, 9:16)*100, 'k-o');
xlim([.8 8.1]); xlabel('Classification')
ylim([0 102])
title('Naive Bayes')
legend( '30%','40%', '50%', '60%','70%', '80%','90%', 'Location', 'NorthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3, 3)
plot(x, results(1, 17:24)*100, 'b-o'); hold on;
plot(x, results(2, 17:24)*100, 'r-o');
plot(x, results(3,17:24)*100, 'g-o');
plot(x, results(4, 17:24)*100, 'm-o');
plot(x, results(5, 17:24)*100, 'c-o');
plot(x, results(6, 17:24)*100, '-o','Color',[.8,.4,.2]);
plot(x, results(7, 17:24)*100, 'k-o');
xlim([.8 8.1]); xlabel('Classification')
ylim([0 102])
title('K Nearest Neighbor')
legend( '30%','40%', '50%', '60%','70%', '80%','90%', 'Location', 'NorthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
```