Jonathan Lu (1438364)
HW3 – Song Classification

This project involved taking samples of various songs from various artists across many different genres, and then developing an algorithm to classify the songs based on the genre or band. This project explored several different supervised machine-learning algorithms, namely *k-nearest neighbors* (knn), *naïve-bayes,* and *linear discriminant analysis* (LDA). With these algorithms implemented, one could construct a classification technique that groups music to the right category.

Sec I: Introduction and Overview

SVD allows us to take a bunch of data and reduce it to its principal components, effectively reducing noise and other aspects that may corrupt the data. The data is then projected onto its principal component space. I implemented various machine-learning algorithms to classify the data. Many statistical methods are implemented in machine learning algorithms - independent predictors, probability, normal models, etc. Machine learning can be both supervised and unsupervised. Unsupervised machine learning is when the computer computes and classifies objects without any input beforehand. Whereas supervised machine learning requires the user to first input labels or other important information beforehand. Since this project involves classifying data with already known fields (i.e. the genres/bands), I chose to use supervised machine learning methods. I chose to analyze and classify rap music, classical music, and future house music (a sub-genre of EDM). In conjunction with these methods, I used cross-validation on a training set to test the data in order to find the accuracy of each method I used. Specifically, I used Matlab's crossval() method which used 10-fold cross-validation, where a random 10% was used as a test set against the remaining 90% training set, across 10 iterations.

Sec II: Theoretical Background

SVD is critical to take the data and decompose it such that it can be represented in another, lower rank space. From SVD, you can produce the U, Σ, V matrices from your data matrix. U and V are orthonormal, so they represent the rotations. The V matrix tells us how much each data projects onto the feature space. Σ is a diagonal matrix in which the values of σ (the singular values) tell us the principal components. By plotting the singular values, I could get a feel of how the data is represented (see magnitude of σ_x figure).

I tested 3 supervised machine-learning algorithms to classify data: *k-nearest neighbors* (knn), *naïve-bayes,* and *linear discriminant analysis* (LDA).

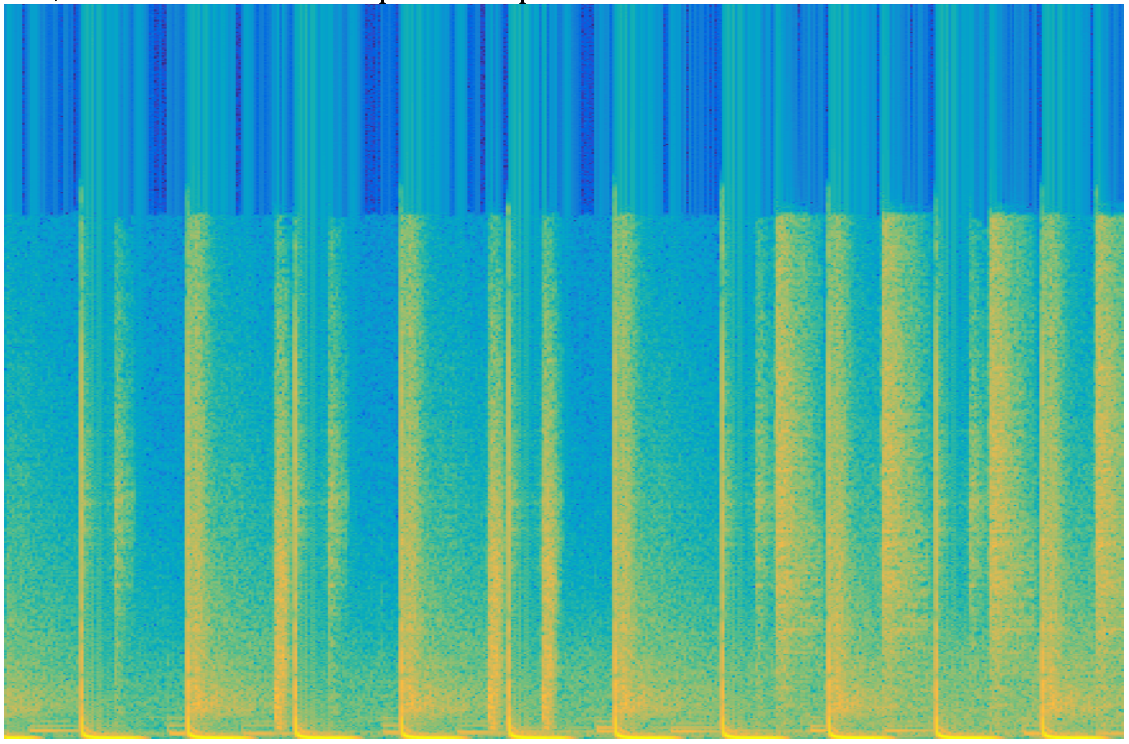
- Knn-search is a method that looks at the k specified nearest neighbors for a new point of data, and that new point of data will be classified based on the largest amount of neighbors of a certain data set.
- Naïve-Bayes is a classifier assumes that all the data is independent, and works based on conditional probabilities.

- LDA involves drawing a line through certain points in space, and then taking the perpendicular displacement from every point, and assuming a normal model identifies certain clusters, where the center peak signifies where the data is most prominent.

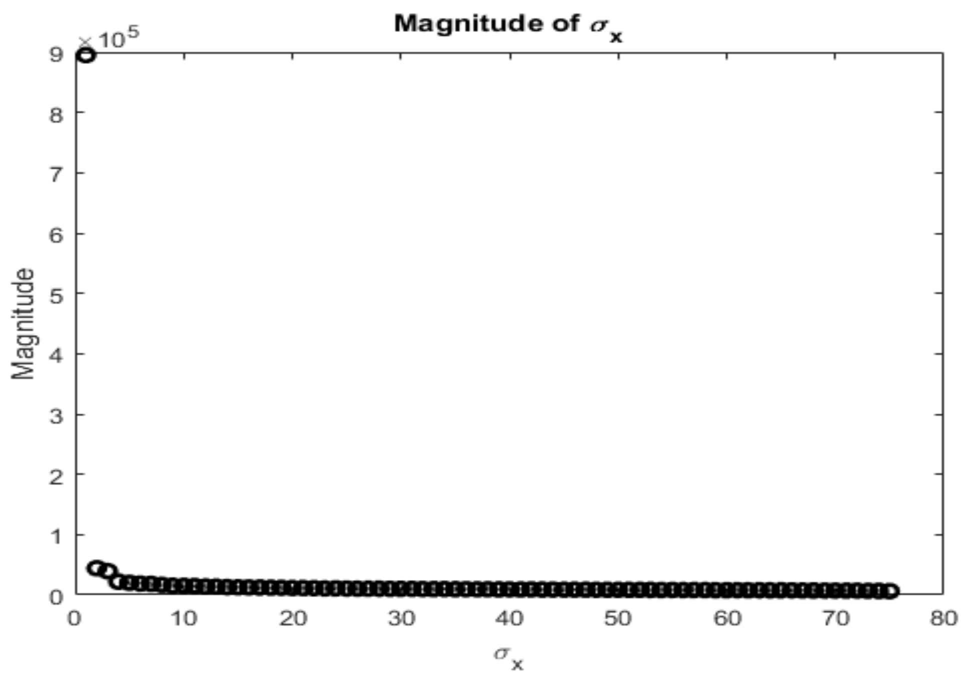
For machine learning to be 'accurate' in a sense, one must employ cross validation. Cross validation is used to create a training set representing a random of the $X\%$ original data, and a testing set representing the remaining $(100 - X)\%$ of the data. Then keep cross validating while keeping track of the accuracy. This is necessary so that if you only run one case, you may get data that does not represent the average.

Sec III: Algorithm Implementation and Development

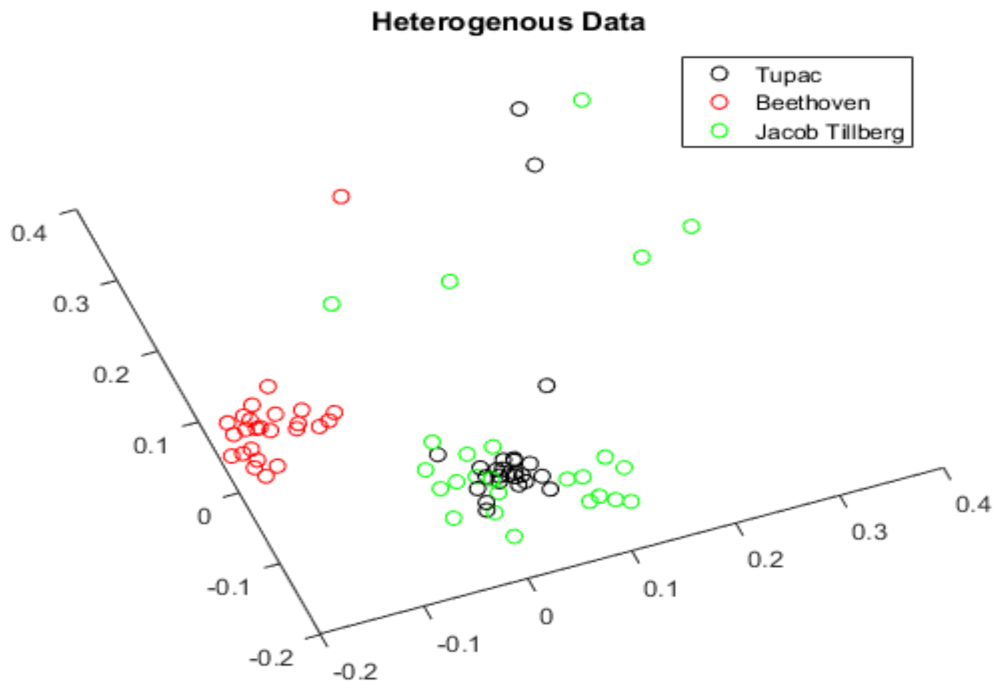
The first step involved processing the audio files. This was done in the processAudioFile() function. I took five second samples of the audio sets, and resampled the same song 5 times across various time spans. With 5 songs per artist, and 3 artists, this constituted 75 samples. I also added 1 song for each artist for the test set, and with 5 samples each, my data matrix had a total of 90 samples (rows). I constructed a spectrogram of the time samples. The spectrogram was based on the frequency of the audio sample over time. An example of what a spectrogram looks like, which varies from sample to sample:



After processing these audio files, they were outputted as a 3-d image matrix. With this image matrix, I converted it into gray scale and also used double() to get double precision accuracy, providing more accurate data. After turning the data from a matrix into a row vector, I added the image into the data matrix and performed SVD on it. With this, I retrieved the U, Σ, V matrices.



Above, we can see the first mode of Σ, which is huge compared to the others. This mode contains a lot of data – where it represents the average "audio" of the data. The modes after describe the variance of the data, which tells us the distinct features of every audio sample. I specifically chose to look at the more dominant modes 2-4, and projected every sample onto the feature space:

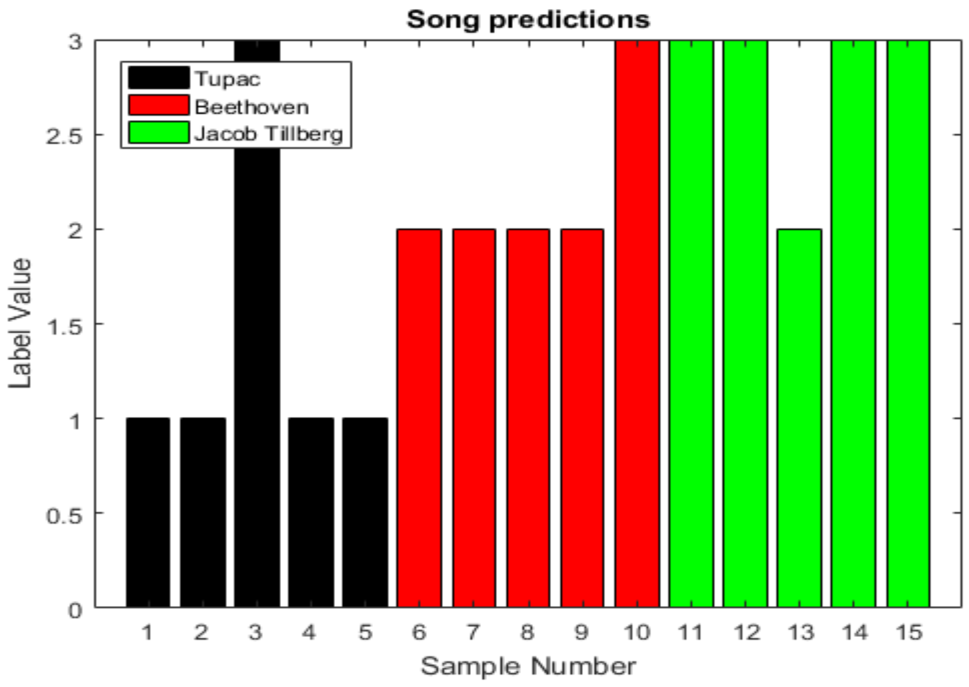


This data signifies how each artist of different genres compare to each other. Initially this data was projected onto a 3D-space, but for ease of viewing I rotated the space onto a 2-D plane in birds-eye view. Note that the Beethoven (classical music) clearly had its own space, besides a few outliers. However, Tupac (rap music) and Jacob Tillberg (Future House music) nearly shared spaces in the feature space. This became an obvious problem in music identification for the machine learning algorithms. There would be a higher error in determining the correct data since the feature space shared was on top of one another, and if new test data landed on top of this space, the computer may have trouble identifying which one was which.

Sec IV: Computational Results

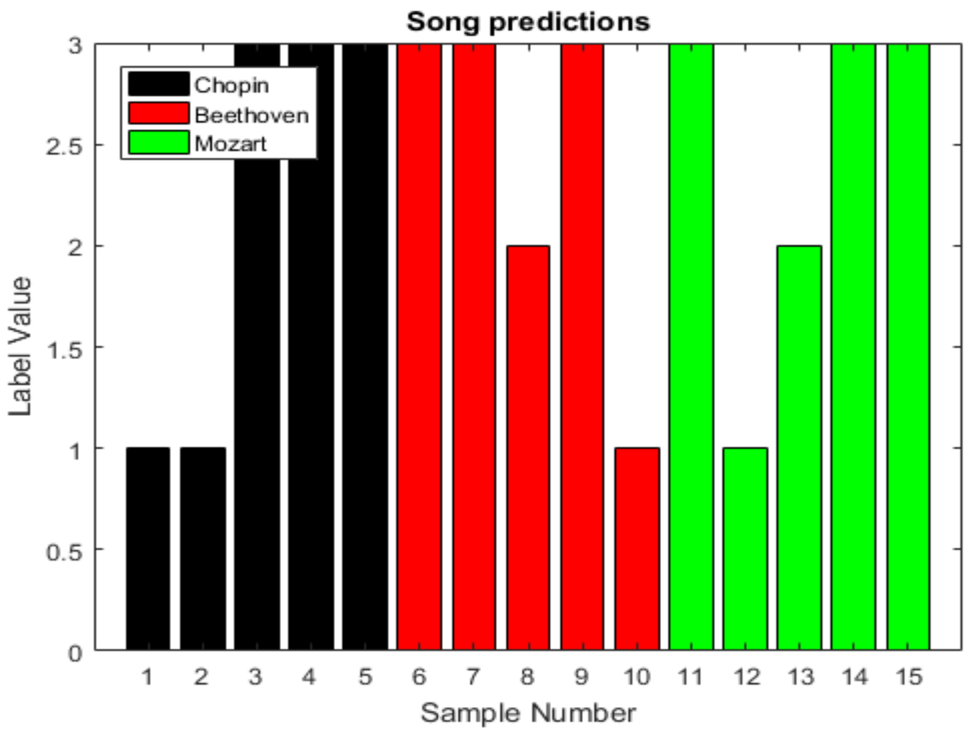
Test 1:

I couldn't do Support Vector Machines (SVM) because the rap music (black data) had its data throughout, and the FH music (green data) overlapped a lot with the black circles. Creating a circular space to enclose the area using SVM would be unwise, even though the majority of the rap music lied in a circular space. So instead I chose Naïve-Bayes. The results are displayed on the bar graph below (I will explain the results in Sec V).

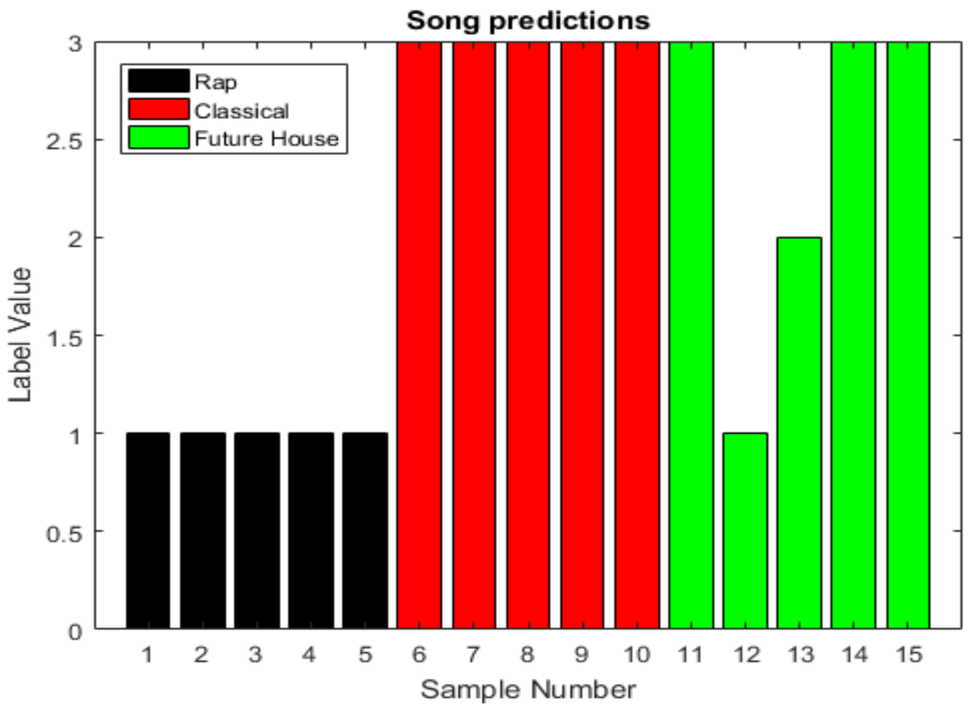


Test 2:

For test 2, I used knn-search for my data. Since test 2 chose to classify bands based on music of the same genre, this test was significantly harder to calculate and had much higher error. The results are displayed on the bar graph below:



<u>Test 3:</u>
I chose to use linear-discriminant analysis (LDA) for test 3. This test involved classifying music and bands based on their genre, and was more accurate then test 2. The bands I chose were Tupac, Kendrick Lamar, and Eminem for rap; Chopin, Beethoven, and Mozart for Classical; and Jacob Tillberg, Retrovision, and MØ for Future House. The results were displayed on a bar graph below:



Sec V: Summary and Conclusions

Note: The height of the bar graph corresponds to the label (i.e. Samples 1-5 are classified by label 1, Samples 6-10 are classified by label 2, and Samples 11-15 are classified by label 3). In a perfect case with 100% accuracy, these test samples should have the black bars of height 1, red bars of height 2, and green bars of heights 3.

Test 1:
This was perhaps the easiest case to test. Given distinct bands of distinct music genres, there was relatively high accuracy using Matlab's kfoldLoss() command on the cross-validated tests. The loss was ~0.2, meaning the accuracy was ~0.8. Looking, back at the Heterogeneous Data (Sec III), I expected most of the loss was on classifying Tupac versus Jacob Tillberg. And the computer misclassified 4 out of 5 of Jacob Tillberg's song as Tupac, and 1 out of 5 of Tupac's songs as Jacob Tillberg's. Interestingly, one of Beethoven's songs also got mistakenly misclassified as Jacob Tillberg's song. This could be due to the outlier data fields far from the average.

Test 2:
The scenario here was the worst case, where the loss was ~0.6, accuracy ~0.4. Given as we were attempting to identify the artist in the same genre of music, this became very difficult for the computer to do accurately, since the frequency of the same music genre tends to be very similar. There were fluctuations for all 3 composers. Chopin had 3/5 samples misclassified, Beethoven had 4/5 samples misclassified, and Mozart had 2/5 samples misclassified.

Test 3
This case was an interesting one. All rap songs were classified correctly, but all classical songs were classified incorrectly. The future house music has 2/5 songs misclassified. Based on the heterogeneous data, classical should have been the easier to classify. But perhaps because there were multiple artists for each genre, the feature space for song samples was simply more spread out. Thus there was no clear line drawn for LDA to perform well on. However, trying the other supervised learning algorithms, LDA performed the best out of all the others. There was ~0.7 accuracy from using kfoldLoss().

Each machine-learning algorithm has its own unique way of identifying and classifying data. Thus, each algorithm performs better under certain situations. It is up to the programmer to know their data in order to decide which algorithm works best.

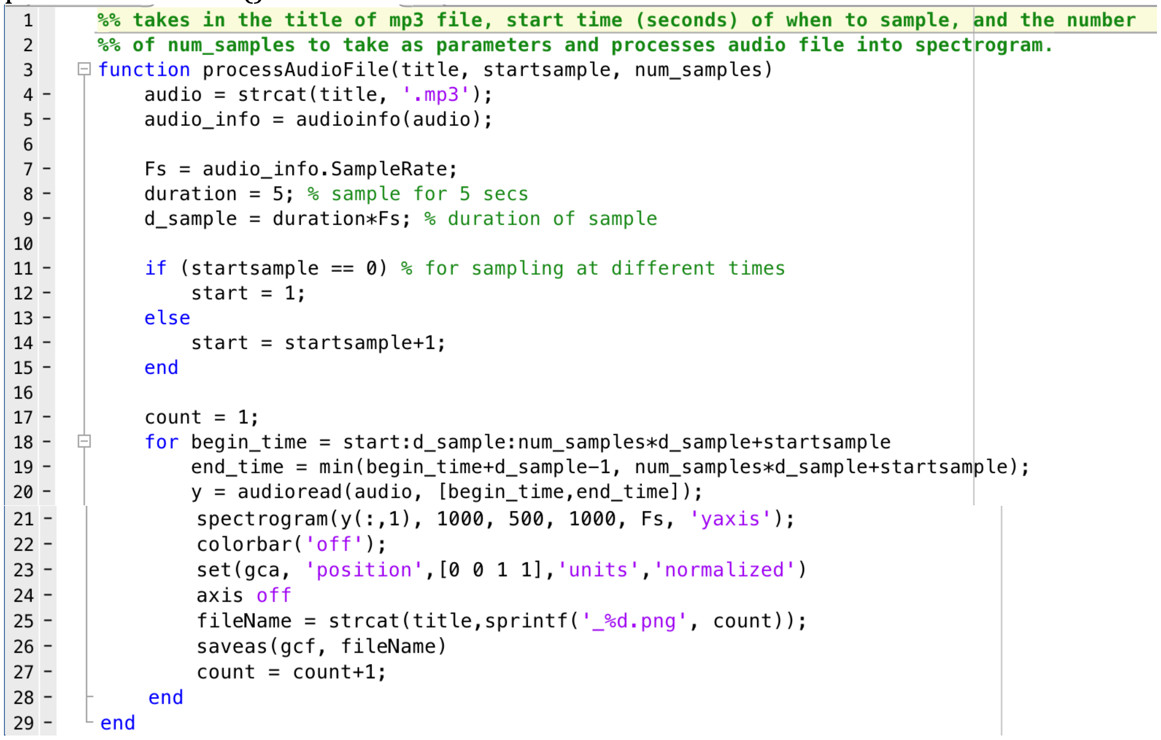
Appendix A: Important Matlab Functions

audioinfo() – read audio file
spectrogram() – used to produce spectrogram of frequency vs time for audio file
rgb2gray() – turns pixels from RGB values to gray scale
double() – turn matrix from uint8 to double precision
reshape() – used to reshape sample data matrix to row vector to add into data matrix

svd() –singular value decomposition to compute U, Σ, V
fitcknn() – perform knn search, specified by numNeighbors = 4
fitcnb() – perform naïve-bayes search
fitcdiscr() – perform linear discrimination analysis
predict() – predict data type for test data
crossval() – perform 10-fold cross validation on training terms
kfoldLoss() – assess the amount of accuracy lost from performing crossval()

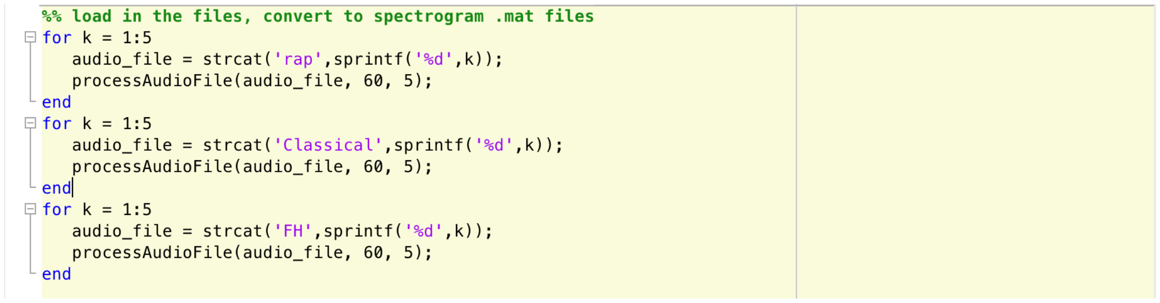
Appendix B: Code

processAudioFile()

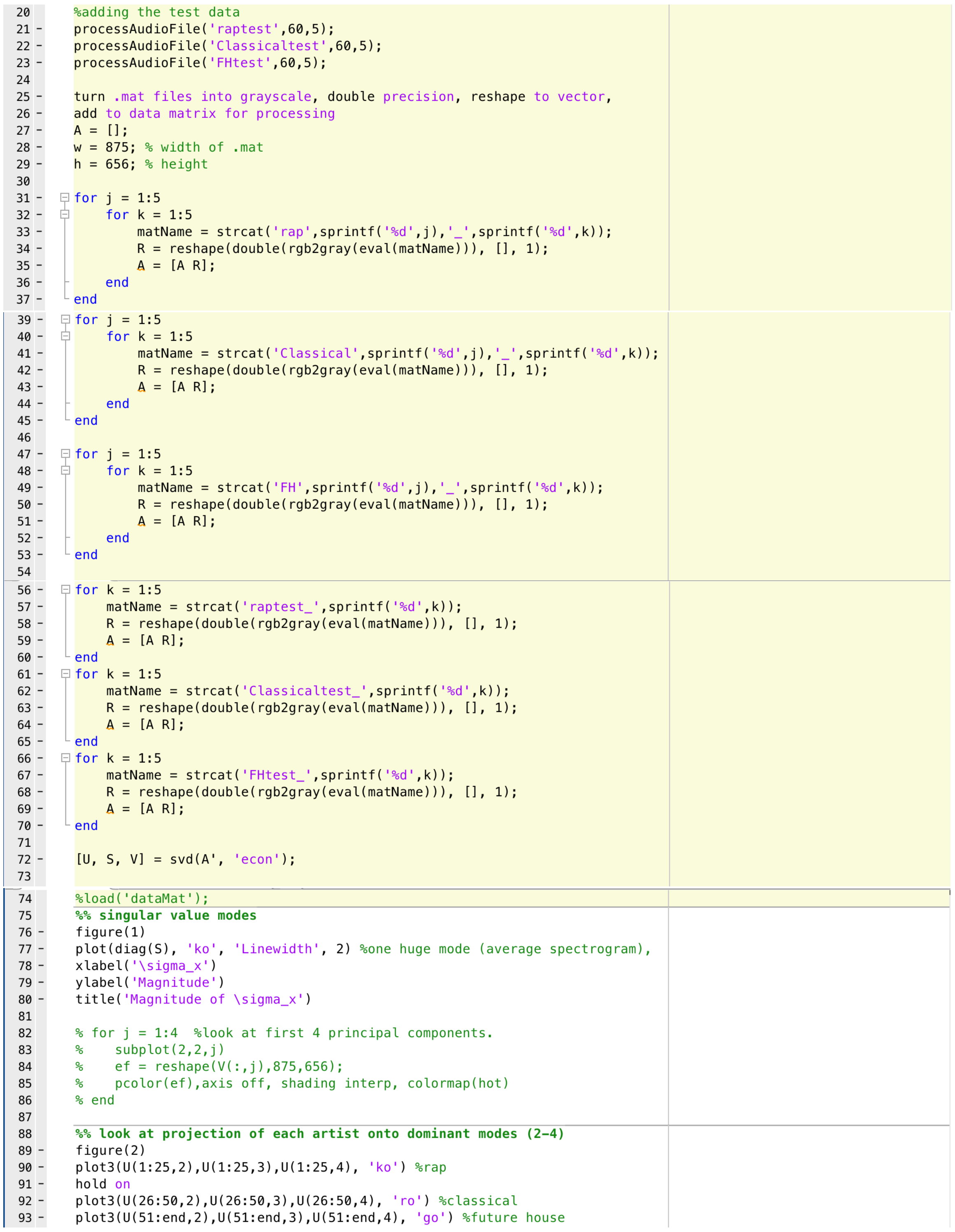


```
%% takes in the title of mp3 file, start time (seconds) of when to sample, and the number
%% of num_samples to take as parameters and processes audio file into spectrogram.
function processAudioFile(title, startsample, num_samples)
    audio = strcat(title, '.mp3');
    audio_info = audioinfo(audio);

    Fs = audio_info.SampleRate;
    duration = 5; % sample for 5 secs
    d_sample = duration*Fs; % duration of sample

    if (startsample == 0) % for sampling at different times
        start = 1;
    else
        start = startsample+1;
    end

    count = 1;
    for begin_time = start:d_sample:num_samples*d_sample+startsample
        end_time = min(begin_time+d_sample-1, num_samples*d_sample+startsample);
        y = audioread(audio, [begin_time,end_time]);
        spectrogram(y(:,1), 1000, 500, 1000, Fs, 'yaxis');
        colorbar('off');
        set(gca, 'position',[0 0 1 1],'units','normalized')
        axis off
        fileName = strcat(title,sprintf('_%d.png', count));
        saveas(gcf, fileName)
        count = count+1;
    end
end
```

main file:



```
%% load in the files, convert to spectrogram .mat files
for k = 1:5
   audio_file = strcat('rap',sprintf('%d',k));
   processAudioFile(audio_file, 60, 5);
end
for k = 1:5
   audio_file = strcat('Classical',sprintf('%d',k));
   processAudioFile(audio_file, 60, 5);
end
for k = 1:5
   audio_file = strcat('FH',sprintf('%d',k));
   processAudioFile(audio_file, 60, 5);
end
```



```matlab
%adding the test data
processAudioFile('raptest',60,5);
processAudioFile('Classicaltest',60,5);
processAudioFile('FHtest',60,5);

turn .mat files into grayscale, double precision, reshape to vector,
add to data matrix for processing
A = [];
w = 875; % width of .mat
h = 656; % height

for j = 1:5
    for k = 1:5
        matName = strcat('rap',sprintf('%d',j),'_',sprintf('%d',k));
        R = reshape(double(rgb2gray(eval(matName))), [], 1);
        A = [A R];
    end
end
for j = 1:5
    for k = 1:5
        matName = strcat('Classical',sprintf('%d',j),'_',sprintf('%d',k));
        R = reshape(double(rgb2gray(eval(matName))), [], 1);
        A = [A R];
    end
end

for j = 1:5
    for k = 1:5
        matName = strcat('FH',sprintf('%d',j),'_',sprintf('%d',k));
        R = reshape(double(rgb2gray(eval(matName))), [], 1);
        A = [A R];
    end
end

for k = 1:5
    matName = strcat('raptest_',sprintf('%d',k));
    R = reshape(double(rgb2gray(eval(matName))), [], 1);
    A = [A R];
end
for k = 1:5
    matName = strcat('Classicaltest_',sprintf('%d',k));
    R = reshape(double(rgb2gray(eval(matName))), [], 1);
    A = [A R];
end
for k = 1:5
    matName = strcat('FHtest_',sprintf('%d',k));
    R = reshape(double(rgb2gray(eval(matName))), [], 1);
    A = [A R];
end

[U, S, V] = svd(A', 'econ');

%load('dataMat');
%% singular value modes
figure(1)
plot(diag(S), 'ko', 'Linewidth', 2) %one huge mode (average spectrogram),
xlabel('\sigma_x')
ylabel('Magnitude')
title('Magnitude of \sigma_x')

% for j = 1:4  %look at first 4 principal components.
%    subplot(2,2,j)
%    ef = reshape(V(:,j),875,656);
%    pcolor(ef),axis off, shading interp, colormap(hot)
% end

%% look at projection of each artist onto dominant modes (2-4)
figure(2)
plot3(U(1:25,2),U(1:25,3),U(1:25,4), 'ko') %rap
hold on
plot3(U(26:50,2),U(26:50,3),U(26:50,4), 'ro') %classical
plot3(U(51:end,2),U(51:end,3),U(51:end,4), 'go') %future house
```

```matlab
%% cross validation
xRap = U(1:25, 2:4);
xClassical = U(26:50, 2:4);
xFH = U(51:75, 2:4);

%% knn-search (2)
X = [xRap; xClassical; xFH]; %data
C = [ones(25,1); 2*ones(25,1); 3*ones(25,1)]; %labels
md1 = fitcknn(X, C, 'NumNeighbors', 4);
md1_cv = crossval(md1); %10-fold training algorithm for cross validation
kloss = kfoldLoss(md1_cv); %see accuracy of program
X_test = [U(76:80, 2:4);U(81:85, 2:4);U(86:90, 2:4)];
pred = predict(md1, X_test);
bar(1:5, pred(1:5),'k')
hold on
bar(6:10, pred(6:10), 'r')
bar(11:15, pred(11:15), 'g')
set(gca, 'Xtick', 1:15)
title('Song predictions')
xlabel('Sample Number')
ylabel('Label Value')
legend('Chopin', 'Beethoven', 'Mozart', 'location' ,'northwest')

%% Naive-Bayes supervised classification (1)
nb = fitcnb(X, C);
cval = crossval(nb);
kloss = kfoldLoss(cval); %0.8
pred = nb.predict(X_test);
bar(pred)
bar(1:5, pred(1:5),'k')
hold on
bar(6:10, pred(6:10), 'r')
bar(11:15, pred(11:15), 'g')
set(gca, 'Xtick', 1:15)
title('Song predictions')
xlabel('Sample Number')
ylabel('Label Value')
legend('Tupac', 'Beethoven', 'Jacob Tillberg', 'location' ,'northwest')

%% LDA (Linear Discriminant Analysis) (3)
lda = fitcdiscr(X,C);
cval = crossval(lda);
kloss = kfoldLoss(cval); %0.7
pred = lda.predict(X_test);
bar(pred)
bar(1:5, pred(1:5),'k')
hold on
bar(6:10, pred(6:10), 'r')
bar(11:15, pred(11:15), 'g')
set(gca, 'Xtick', 1:15)
title('Song predictions')
xlabel('Sample Number')
ylabel('Label Value')
legend('Rap', 'Classical', 'Future House', 'location' ,'northwest')

%% SVM (Support Vector Machines)
% Note: can train with very little data (just need support vectors) -
% different from many other algorithms
% can be used for separated data
% svm = fitcsvm(x_train, c_train);
% pred = classificationsvm(svm, x_test);
% bar(pred)

```