Priyanshi Rastogi
AMATH 482

# Music Classification
# By: Priyanshi Rastogi

1. **Abstract:** In this report we are using Singular Value Decomposition (SVD) and Fourier Transform to help us classify different genres/bands using multiple samples of many songs. We train a model to identify a given five second music clip's genre or artists/band name. We feed in all of these samples of songs, with their respective given labels and then see how our statistical model is able to detect and classify "new" samples of songs into the given genres/band categories.

2. **Introduction/Overview:** Humans can pretty easily recognize different genres/different artists of songs, but it is much harder to make a machine think the same way that our brain learns to distinguish between them. In this assignment we identify music genres by looking at how the frequencies are varied/distributed within a certain time frame. We are given three different tests, one in which we feed in samples of songs from 3 different artists of different genres, the second in which we feed in samples of songs of 3 different bands but of the same genre, and the last test being where we feed in samples of songs for 3 different genres in general. We then break down the audio frequencies into time windows, also known as the Short time Fourier Transform, and further analyze this information to correctly detect which genre/artist the song originated from.

3. **Theoretical Background:** Throughout this assignment, we use the SVD to capture and tell us the important information about the samples of the songs. The SVD of a Matrix A (of size mxn) looks like:

$$\mathbf{A} = \boldsymbol{U\Sigma V^T} \quad \ldots \textcircled{1}$$

Where U, is an orthonormal mxm sized matrix, V is an orthonormal nxn sized matrix, and Σ is an mxn diagonal matrix consisting of the singular values of A ranked in decreasing order. In our case, we perform the SVD on our data matrix consisting of the 5 second samples of several songs, each sample in one column. After performing the SVD, we will be able to see that the proper orthogonal modes will be the vectors in U and the corresponding singular values will tell us how important/valuable those POMs are representing those 5 second clips (Kurtz). We use the SVD to create a low dimensional space to represent our 5 second clips. We then use any sample clip's projection onto that low dimensional space to predict which genre/artist the given clip is related to.
In order to perform the SVD, we take use of the Fourier Transform.

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp(-ikx) f(x) dx \qquad \ldots \textcircled{2}$$

**The Fourier Transform** takes a given time signal and turns it into a frequency signal. Since in our problem we want the frequency and time signal in one, we can first use the **Gabor Transform/Filter** to localize both the time and frequency of a signal and then take the Fourier transform of that.

$$\bar{f}_g(t,k) = \int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)\exp(-ik\tau)d\tau = (f, g_{t,k}) \qquad \ldots \textcircled{3}$$

Here the $f(\tau)$ represents the signal just like the Fourier transform and the $g(\tau - t)$ localizes both time and frequency. Using this, we are then able to transform our 5 second audio clips into frequency and time information signals. Thus, we take the SVD of the spectrogram (created by applying the Gabor and Fourier transform) of the audio signal and do our projections to tell us important information about our clips.
Additionally, we also end up solving the generalized eigenvalue problem for the within and between variances, also known as Linear Discriminant Analysis.

$$S_b \omega = \lambda S_w \omega \quad \ldots \textcircled{4}$$

We end up using the largest eigenvalue and the corresponding generalized eigenvector to tell us what we want to project onto since we want to maximize the between variances, hence give us a more define and clear grouping. Here $S_w$ is the within variance (again wanting it to be as minimum as possible to cluster the data more into their own groups) and the $S_b$ is the *between* variance which we want to maximize to get clear and distinct groups. $S_w$ is there because we want to scale the variance of the means by the variances *within* each group.

4. **Algorithm Implementation and Development:** (The algorithm implementation is generally the same for each test, we use test 1 for ex.)
    1. In order to get multiple 5 second samples of several songs, we used the YouTube to mp3 converter to download songs in mp3 format and then used the program Audacity to cut down each song into a random 5 second clip/sample, made sure that the sampling rate was 44100 for each clip, and then read in each .wav file (SEE LINES 2-15).
    2. To obtain the time-frequency information for each clip, we computed the spectrogram of each 5 second clip (which uses the Gabor Transform that we described in section 3.) Before producing the spectrogram of each clip, we cut down the sampling rate (44100) in half and take every other point of the signal of the clip to reduce the amount of data we use (SEE LINES 13-15). When making the spectrogram of each 5 second clip, we reshape the matrix to make each 5 second clip data to be in a single column. (SEE LINES 15-18)
    3. Once we reshaped each 5 second clip into a single column we put it into a matrix (one matrix for each test) by putting each sample clip (a column) adjacent to each other and grouped such that the samples by one artist/genre are right next to each other followed by the samples of another artist/genre and so on. In our case we had 15 songs for each artist/genre with 5622750 rows. (SEE LINES 17-18).
    4. After getting a matrix for each test, we then perform the reduced SVD on that matrix. (SEE LINE 20). Once we have our U's, S's, and V's computed we projected the 5 second clips onto the POMs (using those as our features) and then the corresponding genre/artist as our label. We explored different values of features to see which number would give us the best grouped data with the least amount of overlap. Then our artists were categorized as the first n number of songs of the matrix (since they were grouped together to start off with) and then the next n number of songs of the matrix for the next artist and so on (SEE LINES 204-end).
    5. We then calculate the variance within and between each group by applying the variance formula using our data for each artist/genre. Once those are calculated, we plug it into our (4) section 3. equation to find the eigenvector corresponding to the largest eigenvalue to project it onto that (again since we want to find a projection that

maximizes the between variances and minimize the within variances). We plot each of the projections of each artist as well as the mean of each projection to give us a good idea where the groups (each artist/genre) is centered around (helping us calculate the threshold later). SEE LINES (218-240/end).

6. After plotting the projections onto the eigenvector, we found in step 5. we can tell how our data is ordered and thus get a rough estimate where each artist/genre lies to then further calculate our thresholds. We can calculate the thresholds by seeing where the initial overlaps happen that transition the points from one artist/genre to other, doing our best to ignore any outliers (Figure 1. we can see this clearly because the 3 artists are different colors). We then plot the thresholds as boundary conditions to lie within the boundary ranges of a genre. (SEE LINES 23-26).
7. After the training has been done, we test our model. We pass in 5 second sample clips of new songs that it has never seen before (in Test 3 of possibly different artists with the corresponding genres) and then do a similar process as we did when training our model. We pass in the new song, make a spectrogram for each song, and reshape it as we mentioned above. We then project it onto the U that we obtained by computing the SVD on our training data matrix and then project it onto the eigenvector with the largest eigenvalue of eq. 4 that we found previously as well. Once we've done all of the projecting, we plot the projected test data onto the same previous plot in which we plotted our training data and thresholds. We then compare where the projected test data point lies (within what ranges of the thresholds) and then classify it as being in the genre corresponding to that range, repeating this step for all test data points and calculating our accuracy at the end since we know the true labels of the test data. (SEE LINES 29-59) SEE FIGURE 1., 2.,3.

5. **Computational Results:** After running test one which focuses on combining 5 second clips of 3 different artists of different genres, we were able to see that they varied quite a bit. The black dots represent the mean of the projections of the sorted artists and they are clearly pretty far away from one another in each Test. Figure 1. looks at the plot of each data point onto the projection of the eigenvector with the largest eigenvalue that solves equation 3. After plotting it we were able to find our threshold values and see that, for ex., Lil Tecca's rap songs seem to generally lie to the left of the first threshold, Beethoven's orchestra songs lie to the right of the 2nd threshold, and Miguel's RnB songs that lie between the two thresholds. As we can see from Figure 1. 5/6 samples of test Miguel clips landed within the Miguel range giving our model an 83% accuracy rate of predicting a new RnB song correctly. From Figure 2. We can see that 3/6 samples of Beethoven landed within the corresponding range, giving it an accuracy rate of 50% to predict a new Orchestra song correctly, and finally as we can see from Figure 3. our model was able to predict 5/6 new Lil Tecca songs giving it an 83% accuracy to predict a new Rap song correctly.

   Test 2 was a bit harder to distinguish since, even though we were looking at 3 different artists, they were all of the same genre, thus sounding pretty similar. From Figure 4. We found out that 3/6 samples of Miguel landed within the corresponding range (that was considered to be Miguel) giving it an accuracy rate of 50%. From Figure 4. Weeknd's test projections also had a 50% accuracy rate (3/6 samples correct), and from Figure 5. Frank Ocean's test projections also had a 50% accuracy rate (3/6 samples correct). Thus, as we mentioned above, it makes sense that our accuracy rates are a little lower for Test 2 than compared to Test 1.

Priyanshi Rastogi
AMATH 482

Test 3 was even harder to detect than the first two tests since we fed in different songs from multiple different artists. From Figure 7. Our accuracy rate of success for testing RnB songs was 3/5 songs (or 60%), for testing Rap was 2/5 (40%), and for testing EDM was 2/5 songs (40%) again.

**(note: Legend didn't seem to work correctly)**

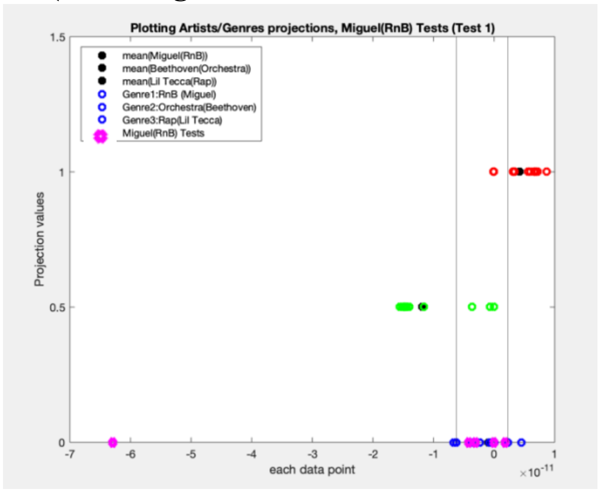

Figure 1: Showing RnB Projected Test Points with original training projected data points & thresholds. Test 1, Blue-Miguel, Red-Beethoven,Green-Rap)

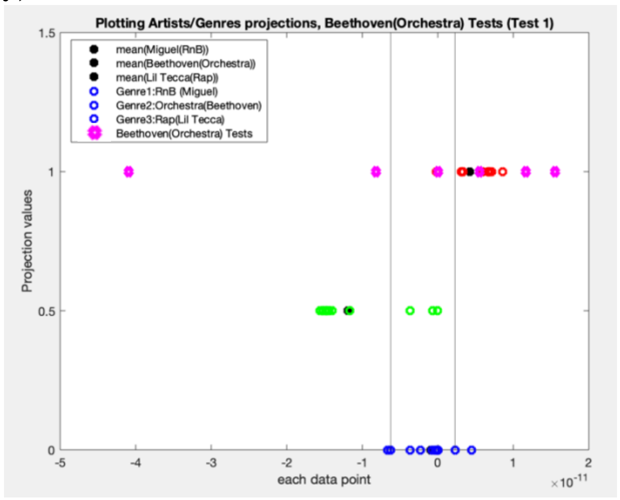

Figure 2: Showing Orchestra songs projected Test values with original training projected data points & thresholds (Test 1, Blue-Miguel, Red-Beethoven,Green-Rap)

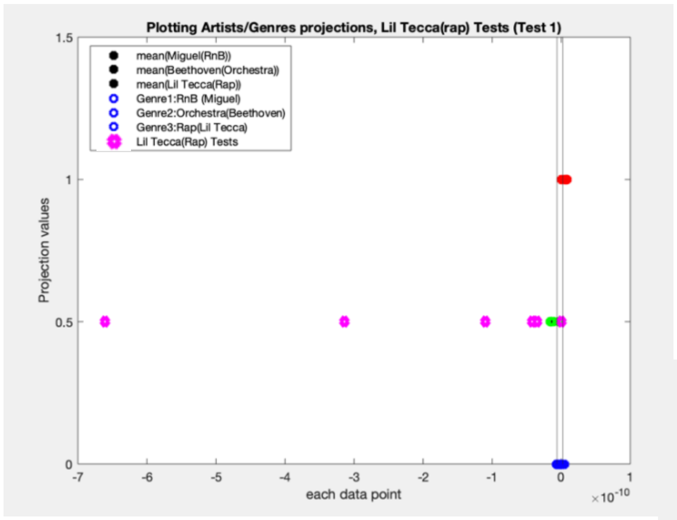

Figure 3: Showing Rap songs Projected Test values with original training projected data points & thresholds (Test 1, Blue-Miguel, Red-Beethoven,Green-Rap)

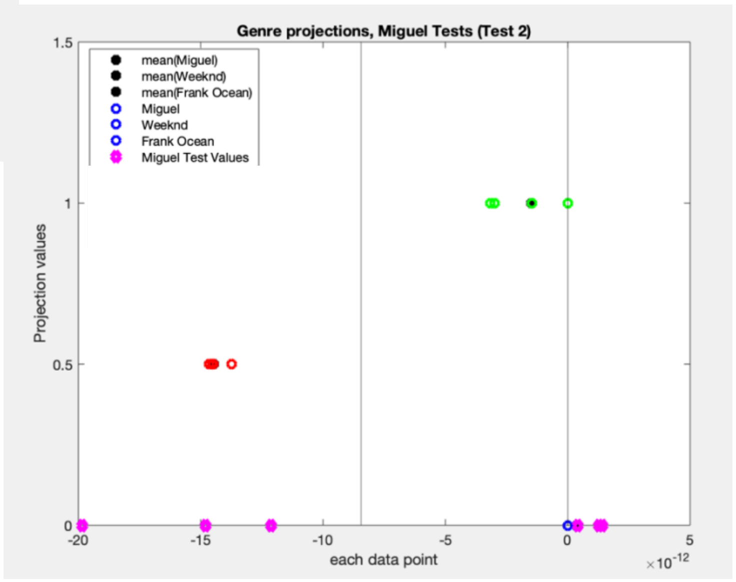

Figure 4: Showing the RnB Genre Projections with Trained Points for each artist, thresholds, and Miguel's songs projected test values in pink. (Test 2, blue-Miguel, red-Weeknd, Blue-Frank Ocean)

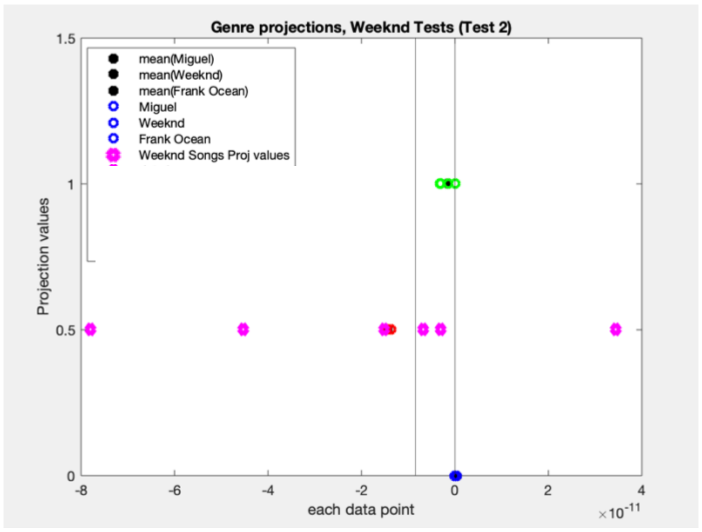

Figure 5: Showing the RnB Genre Projections with Trained Points for each artist, thresholds, and Weeknd's songs' projected test values in pink. (Test 2, blue-Miguel, red-Weeknd, Blue-Frank Ocean)

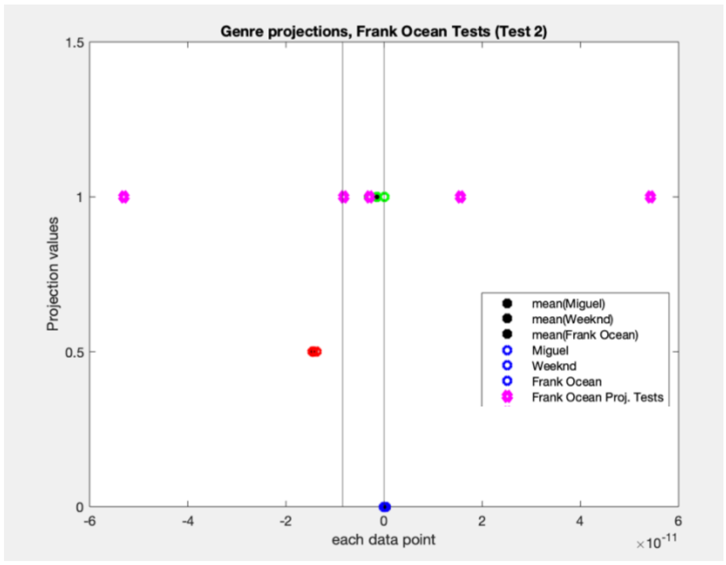

Figure 6: Showing the RnB Genre Projections with Trained Points for each artist, thresholds, and Frank Ocean's songs projected test values in pink. (Test 2, blue-Miguel, red-Weeknd, Blue-Frank Ocean)

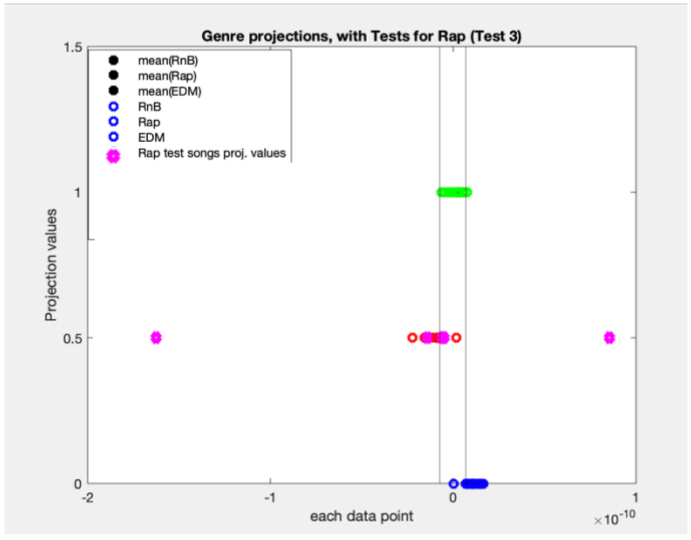

Figure 7: Showing the Genre Projections with Trained Projections for each artist, thresholds, and RnB songs' projected test values in pink. (Test 3, blue-RnB, red-Rap, Blue-EDM)

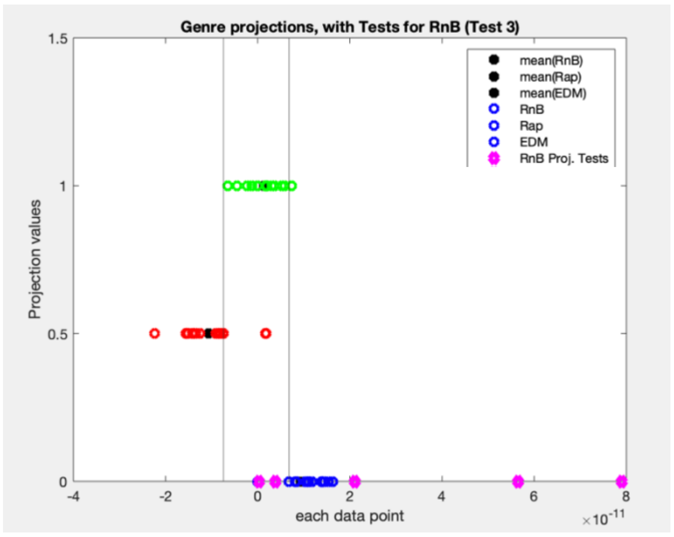

Figure 8: Showing the Genre Projections with Trained Projections for each artist, thresholds, and Rap test songs' projected test values in pink. (Test 3, blue-RnB, red-Rap, Blue-EDM)

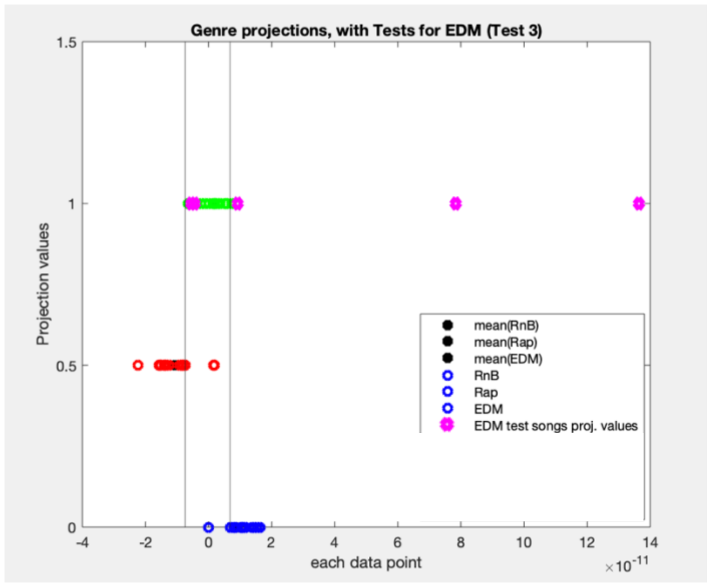

Figure 9: Showing the Genre Projections with Trained Projections for each artist, thresholds, and EDM test songs' projected test values in pink. (Test 3, blue-RnB, red-Rap, Blue-EDM)

## 6. Summary and Conclusions:

In summary we were able to notice how computing and using the SVD can give us a lower dimension representation of our data. We can use that to train a machine learning model to identify artists or genres. We can see that overall, Test 1 did the best in performance. This makes logical sense because we pass in songs *only* from three different artists each of whose genres are *different* from each other. For Test 1 we ended up getting a success rate of about 83% for two artists/genres and 50% for the third. Test 2 was a bit harder because even though we chose 3 different artists, they were each from the same genre, thus the music sounded much more similar in comparison to Test 1. For Test 2, our model was able to detect each artist with a 50% success rate. Finally, test 3, was our most difficult test since we were passing in different songs from different artists relating to 3 different genres, with about a 40% success rate. We randomly sampled 5 seconds of a song which likely affected the accuracy of the classifier (possibly less accurate). To make it even more accurate we could use more training data, thus even further generalizing the features we use for each genre/artist.

## 7. Appendix A-Function Descriptions:

- plot(x,y): plots passed in parameters x over y
- zeros(n,m): creates a matrix of size nxm with all zeros as its values.
- length(v): returns the length of the passed in parameter
-size(x): Gives you the dimension of the passed in parameter
-audioread('x.wav'): Reads the passed in .wav file
-svd: perform svd on the passed in matrix
-reshape: reshapes passed in matrix into nxm size you give it.

-eig: computes the eigenvalues of the matrix and can return the eigenvectors of the matrix as well as a diagonal matrix consisting of the eigenvalues.
-max: finds the maximum element in the passed in matrix/vector.

## 8. Appendix B:

```
%Test 1:
songs = {'adorn.wav','sureThing.wav', 'miguel3.wav','miguel4.wav','miguel5.wav',...
'miguel6.wav','miguel7.wav','miguel8.wav','miguel9.wav','miguel10.wav',...
'miguel11.wav','miguel12.wav','miguel13.wav','miguel14.wav','miguel15.wav',...
'fur_elise.wav','beethoven2.wav','beethoven3.wav','beethoven4.wav','beethoven5.wav',...
'beethoven6.wav','beethoven7.wav','beethoven8.wav','beethoven9.wav','beethoven10.wav',...
'beethoven11.wav','beethoven12.wav','beethoven13.wav','beethoven14.wav','beethoven15.wav',...
'ransom.wav','loveMe.wav','lilTecca3.wav','lilTecca4.wav','lilTecca5.wav','lilTecca6.wav',...
'lilTecca7.wav','lilTecca8.wav','lilTecca9.wav','lilTecca10.wav','lilTecca11.wav',...
'lilTecca12.wav','lilTecca13.wav','lilTecca14.wav','lilTecca15.wav'};
    test1Matrix = zeros(5622750,length(songs));
    for k = 1:length(songs)
        [y,Fs] = audioread(songs{k});
        Fs = Fs/2;
        y = y(1:2:length(y));
        [spectrogramMatrix] = makeSpectrogram(y,Fs);
        spectrogramMatrix = reshape(spectrogramMatrix,5622750,1);
        test1Matrix(:,k) = spectrogramMatrix(:,1);
    end
    [U1,S1,V1,miguel,beethoven,lilTecca,sortedMiguel,sortedBeethoven,...
        sortedLilTecca,w] = trainer(test1Matrix,30,songs);
    %calcuating thresholds basedo on the order of the projections
    threshold1 = sortedMiguel(2); %based on plot
    threshold2 = sortedMiguel(t2-1);
    xline(threshold1)
    xline(threshold2)
    hold on
    %Testing: feeding in new 6 different clips per artist
    testSongs = {'testMiguel1.wav','testMiguel2.wav','testMiguel3.wav','testMiguel4.wav','testMiguel5.wav',...
'testMiguel6.wav','testBeethoven1.wav','testBeethoven2.wav','testBeethoven3.wav','testBeethoven4.wav'...
'testBeethoven5.wav','testBeethoven6.wav','testLilTecca1.wav','testLilTecca2.wav','testLilTecca3.wav',...
        'testLilTecca4.wav','testLilTecca5.wav','testLilTecca6.wav'};
    for k = 1:length(testSongs)
        [y,Fs] = audioread(testSongs{k});
        Fs = Fs/2;
        y = y(1:2:length(y));
        [spectrogramMatrix] = makeSpectrogram(y,Fs);
        spectrogramMatrix = reshape(spectrogramMatrix,5622750,1);
        testSong = U1'*spectrogramMatrix;
        projTestSong = w'*testSong;
        disp(k)
        if ((projTestSong <= threshold2) && (projTestSong>=threshold1))
            disp('Miguel')
        elseif (projTestSong> threshold2)
            disp('Beethoven')
        else
            disp('LilTecca')
        end
        if k <=6 %checking the Miguel test songs
            %plot(projTestSong, 0, 'dm','Linewidth', 9)%blue for Miguel
        elseif k>6 && k<=12
            %plot(projTestSong, 1,'dm','Linewidth', 9)%red for Beethoven
        else
            %plot(projTestSong, .5, 'dm','Linewidth', 9)%lilTecca green
        end
    end
    ylim([0 1.5])
    title('Plotting Artists/Genres projections, Miguel(RnB) Tests (Test 1)')
    legend('mean(Miguel(RnB))','mean(Beethoven(Orchestra))','mean(Lil Tecca(Rap))','Genre1:RnB (Miguel)','Genre2:Orchestra(Beethoven)','Genre3:Rap(Lil Tecca)','Miguel(RnB) Tests','','','')
    %% Test 2:
    songs = {'adorn.wav','sureThing.wav', 'miguel3.wav','miguel4.wav','miguel5.wav',...
'miguel6.wav','miguel7.wav','miguel8.wav','miguel9.wav','miguel10.wav',...
```

```
'weeknd1.wav','weeknd2.wav','weeknd3.wav','weeknd4.wav','weeknd5.wav',...
'weeknd6.wav','weeknd7.wav','weeknd8.wav','weeknd9.wav','weeknd10.wav',...
'ocean1.wav','ocean2.wav','ocean3.wav','ocean4.wav','ocean5.wav','ocean6.wav'...
         'ocean7.wav','ocean8.wav','ocean9.wav','ocean10.wav'};
    test2Matrix = zeros(5622750,length(songs));
    for k = 1:length(songs)
        [y,Fs] = audioread(songs{k});
        Fs = Fs/2;
        y = y(1:2:length(y));
        [spectrogramMatrix] = makeSpectrogram(y,Fs);
        spectrogramMatrix = reshape(spectrogramMatrix,5622750,1);
        test2Matrix(:,k) = spectrogramMatrix(:,1);
    end
    [U2,S2,V2,miguel2,weeknd,ocean,sortMiguel2,sortWeeknd,sortOcean,w] = trainer(test2Matrix,25,songs);
    %calculating thresholds, looked at projection plot for the ordering
    t1 = 1;
    t2 = length(sortOcean);
    t3 = 1;
    t4 = length(sortMiguel2);
    while t1<length(sortWeeknd) && sortWeeknd(t1)<sortOcean(t2)
        t1 = t1+1;
        t2 = t2-1;
    end
    while (sortOcean(t3)<sortMiguel2(t4))
        t3 = t3+1;
        t4 = t4-1;
    end
    thr1 = (sortWeeknd(t1)+sortOcean(t2))/2;
    thr2 = (sortOcean(t3)+sortMiguel2(t4))/2;
    xline(thr1)
    hold on
    xline(thr2)
    ylim([0 1.5])
    title('Genre projections, Weeknd Tests (Test 2)')
    legend({'mean(Miguel)','mean(Weeknd)','mean(Frank Ocean)','Miguel','Weeknd','Frank Ocean','Weeknd Songs Proj values'})
    hold on
    %Tests:
    % testSongs2 = {'test2Miguel1.wav','test2Miguel2.wav','test2Miguel3.wav'};
    testSongs2 = {'test2Miguel1.wav','test2Miguel2.wav','test2Miguel3.wav','test2Miguel4.wav',...
```

```
'test2Miguel5.wav','test2Miguel6.wav','test2Weeknd1.wav','test2Weeknd2.wav','test2Weeknd3.wav',....
'test2Weeknd4.wav','test2Weeknd5.wav','test2Weeknd6.wav','test2Ocean1.wav','test2Ocean2.wav',...
'test2Ocean3.wav','test2Ocean4.wav','test2Ocean5.wav','test2Ocean6.wav'};
    for k = 1:length(testSongs2)
        [y,Fs] = audioread(testSongs2{k});
        Fs = Fs/2;
        y = y(1:2:length(y));
        [spectrogramMatrix] = makeSpectrogram(y,Fs);
        spectrogramMatrix = reshape(spectrogramMatrix,5622750,1);
        testSong = U2'*spectrogramMatrix;
        projTestSong = w'*testSong;
    %     if ((projTestSong <= thr2) && (projTestSong>=thr1))%anything between the thresholds
    %         disp('Frank Ocean')
    %     elseif (projTestSong<thr1)
    %         disp('Weeknd')
    %     elseif (projTestSong>thr2)
    %         disp('Miguel')
    %     end
        if k <=6 %checking the Miguel test songs
            %plot(projTestSong, 0, 'dm','Linewidth', 9)%blue for Miguel
        elseif k>6 && k<=12
            plot(projTestSong, .5,'dm','Linewidth', 9)%red for Weeknd
        else
            %plot(projTestSong, 1, 'dm','Linewidth', 9)%Frank Ocean
        end
    end
    %% Test 3:
    songs3 = {'test3Rnb1.wav','test3Rnb2.wav','test3Rnb3.wav','test3Rnb4.wav',...
'test3Rnb5.wav','test3Rnb6.wav','test3Rnb7.wav','test3Rnb8.wav','test3Rnb9.wav',...
'test3Rnb10.wav','test3Rnb11.wav','test3Rnb12.wav','test3Rnb13.wav','test3Rnb14.wav',...
'test3Rnb15.wav','test3Rap1.wav','test3Rap2.wav','test3Rap3.wav','test3Rap4.wav',...
'test3Rap5.wav','test3Rap6.wav','test3Rap7.wav','test3Rap8.wav','test3Rap9.wav',...
'test3Rap10.wav','test3Rap11.wav','test3Rap12.wav','test3Rap13.wav','test3Rap14.wav',...
```

```
        'test3Rap15.wav','test3Edm1.wav','test3Edm2.wav','test3Edm3.wav','test3Edm4.wav',...
        'test3Edm5.wav','test3Edm6.wav','test3Edm7.wav','test3Edm8.wav','test3Edm9.wav',...
        'test3Edm10.wav','test3Edm11.wav','test3Edm12.wav','test3Edm13.wav','test3Edm14.wav',...
        'test3Edm15.wav',};
    test3Matrix = zeros(5622750,length(songs3));
    for k = 1:length(songs3)
        [y,Fs] = audioread(songs3{k});
        Fs = Fs/2;
        y = y(1:2:length(y));
        [spectrogramMatrix] = makeSpectrogram(y,Fs);
        spectrogramMatrix = reshape(spectrogramMatrix,5622750,1);
        test3Matrix(:,k) = spectrogramMatrix(:,1);
    end
    [U3,S3,V3,rnb,rap,edm,sortRnB,sortRap,sortEdm,w] = trainer(test3Matrix,28,songs3);
    %calculating thresholds, looked at projection plot for the ordering
    thr1 = sortRap(length(sortRap)-2);
    thr2 = sortRnB(4);
    xline(thr1)
    hold on
    xline(thr2)
    ylim([0 1.5])
    title('Genre projections, with Tests for EDM (Test 3)')
    legend({'mean(RnB)','mean(Rap)','mean(EDM)','RnB','Rap','EDM','EDM test songs proj. values'})
    hold on
    testSongs3 = {'rnbTest1.wav','rnbTest2.wav','rnbTest3.wav','rnbTest4.wav','rnbTest5.wav',...
    'rapTest1.wav','rapTest2.wav','rapTest3.wav','rapTest4.wav','rapTest5.wav',...
    'edmTest1.wav','edmTest2.wav','edmTest3.wav','edmTest4.wav','edmTest5.wav'};
    for k = 1:length(testSongs3)
        [y,Fs] = audioread(testSongs3{k});
        Fs = Fs/2;
        y = y(1:2:length(y));
        [spectrogramMatrix] = makeSpectrogram(y,Fs);
        spectrogramMatrix = reshape(spectrogramMatrix,5622750,1);
        testSong = U3'*spectrogramMatrix;
        projTestSong = w'*testSong;
        if ((projTestSong <= thr2) && (projTestSong>=thr1))%anything between the thresholds
            disp('EDM')
        elseif (projTestSong<thr1)
            disp('Rap')
        elseif (projTestSong>thr2)
            disp('RnB')
        end
        if k <=5 %checking the RnB test songs
            %plot(projTestSong, 0, 'dm','Linewidth', 9)%blue for RnB
        elseif k>5 && k<=10
            %plot(projTestSong, .5,'dm','Linewidth', 9)%red for Rap
        else
            %plot(projTestSong, 1, 'dm','Linewidth', 9)%green for EDM
        end
    end
    function [specMatrix] = makeSpectrogram(y, Fs)
       v = y.';
       v = (v(1,:) + v(2,:))./2;%averaging of v
       L = 5;
        n = (Fs)*L; %change the frequency scale
        k=(1/L)*[0:(n/2)-1 (-n)/2:-1];
        a = 1;
        t2 = linspace(0,L,n+1);
        t = t2(1:n);
        tslide = 0:0.1:L;
       specMatrix = zeros(length(tslide), n);
            for j = 1:length(tslide)
                g = exp(-a.*((t-tslide(j)).^2)); %the gabor transform
                Vg = g.*v;
                Vgt = fft(Vg);
                specMatrix(j,:) = fftshift(abs(Vgt));
            end
    end

    function [U,S,V,artist1,artist2,artist3,sortArtist1,...
        sortArtist2,sortArtist3,w] = trainer(testMatrix,features,songs)
        [U,S,V] = svd(testMatrix, 'econ');
        artists = S*V'; %projection onto V (the principal components)
        n1 = length(songs)/3;n2 = length(songs)/3;n3 = length(songs)/3;
        U = U(:, 1:features);
        artist1 = artists(1:features, 1:n1);
        artist2 = artists(1:features,(n1+1):(n1+n2));
        artist3 = artists(1:features, (n1+n2+1):(n1+n2+n3));
        mArtist1 = mean(artist1,2);
        mArtist2 = mean(artist2,2);
        mArtist3 = mean(artist3,2);
        mTotal = mean(artists(1:features,:),2);
```

```matlab

        Sw = 0; %formula to calculate the variance within each group
        for k = 1:n1
            Sw = Sw + (artist1(:,k)-mArtist1)*(artist1(:,k)-mArtist1)';
        end
        for k = 1:n2
            Sw = Sw + (artist2(:,k)-mArtist2)*(artist2(:,k)-mArtist2)';
        end
        for k = 1:n3
            Sw = Sw + (artist3(:,k)-mArtist3)*(artist3(:,k)-mArtist3)';
        end
         SbArtist1 = (mArtist1-mTotal)*(mArtist1-mTotal)';
         SbArtist2 = (mArtist2-mTotal)*(mArtist2-mTotal)';
         SbArtist3 = (mArtist3-mTotal)*(mArtist3-mTotal)';
         Sb = SbArtist1+SbArtist2+SbArtist3;

    %finding the eigenvector corresponding to the largest eigenvalue
        [V2, D] = eig(Sb, Sw);
        [~,ind] = max(abs(diag(D)));
        w = V2(:,ind); w = w/norm(w,2);
        %projecting onto 1st eigenvector (w/largest e-value)
        vArtist1 = w'*artist1;
        vArtist2 = w'*artist2;
        vArtist3 = w'*artist3;
        sortArtist1 = sort(vArtist1);
        sortArtist2 = sort(vArtist2);
        sortArtist3 = sort(vArtist3);
        plot(mean(vArtist1), 0, '*k','Linewidth',8)
        hold on
        plot(mean(vArtist2), .5, '*k','Linewidth',8)
        hold on
        plot(mean(vArtist3), 1, '*k','Linewidth',8)
        hold on
         plot(vArtist1, zeros(n1),'ob','Linewidth',2)%RnB
         hold on
         plot(vArtist2,0.5*ones(n2),'or','Linewidth',2)%Rap
         hold on
         plot(vArtist3,ones(n3),'og','Linewidth',2)%Edm
         hold on
         ylabel('Projection values')
         xlabel('each data point')
         ylim([0,2])
    End
```

**References:** Kutz, Nathan J. (2013). *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data.*