Music Genre Identification

Nick Moore

February 20, 2017

Abstract

Predictive modeling is an important application of data analysis techniques, and is a current hot topic in the field of data science. Herein a predictive model for music artist/genre classification is created. Five second samples of songs from three different artists/genres undergo singular value decomposition and the resultant matrices are with Naïve Bayes classification algorithm to create a model that classifies the samples by artist/genre. Three tests were conducted, the first comparing three artists form different genres, the second comparing three artists from the same genre, and the third comparing multiple artists across three different genres.

Sec. I. Introduction & Overview

The goal of this report is to explore clustering and classification methods with a real data set. Songs from a variety of artists and genres make up the real data set, with three different test cases driving the report. For the first test case three different artists were chosen from three distinct genres. I chose the Beatles, Snoop Dogg, and David Bowie. The genres covered are rock, rap, and pop, respectively. The second test case analyzed three different artists of the same genre, the Beatles, the Rolling Stones, and Jimi Hendrix, all indie rock artist. The third and final test case compared multiple artists in three different genres. Rock, rap, and pop were compared with three artists for each genre. For tests one and two 80 songs from each artist were selected, and for test case three a total of 60 songs per genre (20 songs per artist) were selected. The spectrograms of the samples were taken and reshaped into a new matrix X where each row vector was a different sample. The matrix X then underwent singular value decomposition to generate matrices U, Σ, and V, where U contains the principal components, Σ contains the variance captured by the principal components, and V contains the projection of each song sample onto the principal components. From here samples were randomly assigned to the training and test sets, with 80% being used in the training set. Naïve bayes was used to classify the data and cross validation was performed. The cross validation was performed for 50 iterations and used to determine the percent success of each test case.

Sec. II. Theoretical Background

The spectrogram of the songs are taken and used for analysis. A spectrogram is a three dimensional representation of sound with the three dimensions being time, frequency, and amplitude. An example spectrogram is shown in figure 1. These three dimensional matrices are converted in row vectors and stacked to form matrix X, which undergoes singular value decomposition and Naïve Bayes classification. Naïve Bayes is a supervised classification algorithm that is built upon Bayes theorem. The equation below shows the basis of this algorithm where P(1|x) and P(0|x) are the probability of being in class one or zero given data x, P(x|1) and P(x|0) are the probability of x given one or zero, and P(1) and P(0) are the probability distributions of one and zero. The right side of the above equation is relatively easy to calculate, so P(1|x) and P(0|x) are able to be determined from this. These conditional probabilities are used as the basis of the classification algorithm.[1]

$$\frac{P(1|x)}{P(0|x)} = \frac{P(x|1)P(1)}{P(x|0)P(0)}$$

Figure 1: Spectrogram of sample from *Gin and Juice* by Snoop Dogg

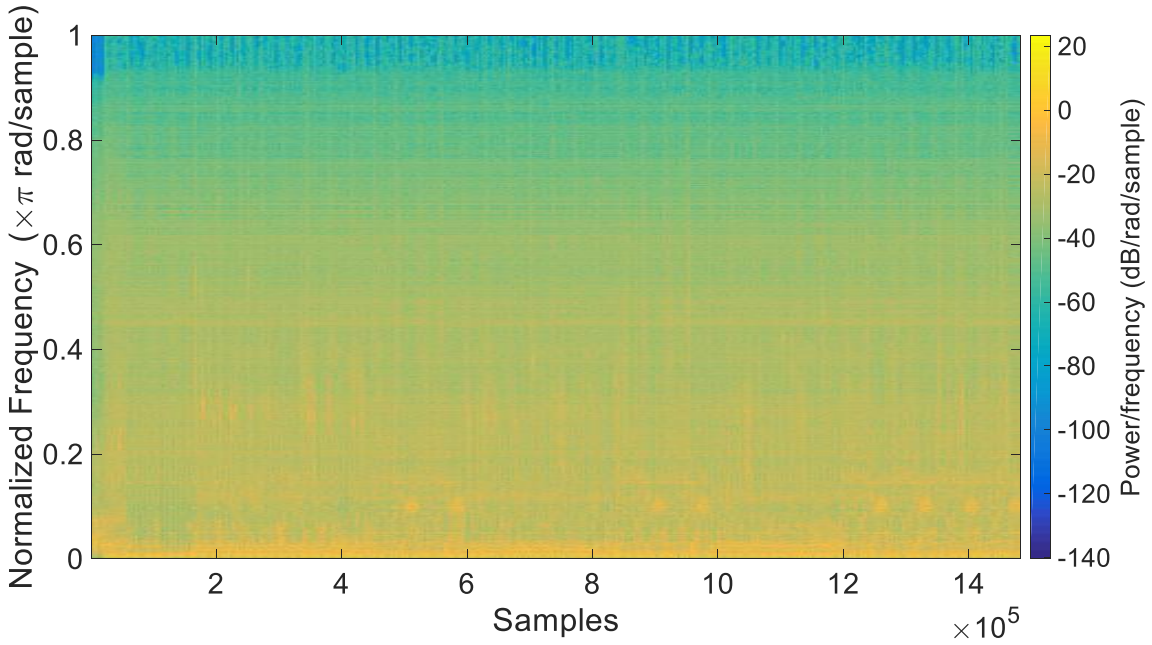

Figure 1 shows the spectrogram of the 5 second sample of Gin and Juice by Snoop Dogg. The 5 second sample is taken from the 60 to 65 second timeframe of the song.

Sec III. Algorithm Implementation & Development

The first step of the music identification process was to convert the song files into data matrices for MATLAB computations. One function, MusicConvert, was used to do this (lines 301-310). MusicConvert took an mp3 file as the input and output a spectrogram of a 5 second sample of the song. To do this first the average of the two stereo signals from the mp3 file was taken. A 5 second segment was cropped out and converted to a spectrogram. Samples were taken at the one minute mark for all of the songs. This resulted in a total of 240 samples for tests one and two, and 180 samples for test 3. For all three test cases 80% were used for the training set and the remaining 20% were used for the test set. The spectrograms of the 5 second samples were reshaped to form row vectors that were stacked in the matrix X. SVD was applied to the transpose of matrix X to generate matrices U, Σ, and V (lines 108-117). Matrix V contains information regarding the spectrogram's projections onto the principal components (lines 119-121). With matrix V separated into the different artist/genre groups the training and test sets were created. Random permutations were used to assign samples to the training and test sets for cross validation (lines 130-140). Naïve Bayes was used for the classification (lines 132-133). Cross validation was repeated 50 times for each test and the percentage of correct identification for the test cases and each artist/genre was calculated (lines 128). Table 1 and figures 2-4 show these results for the three test cases.

Sec IV. Computational Results

Music genre identification comes easily to people, so herein I explored how well different artist/genres could be identified using a Naïve Bayes classification method. The first step of the analysis was to generate the spectrograms. An example spectrogram from the 5 second segment of Snoop

Figure 2: Variance Captured by σ_n

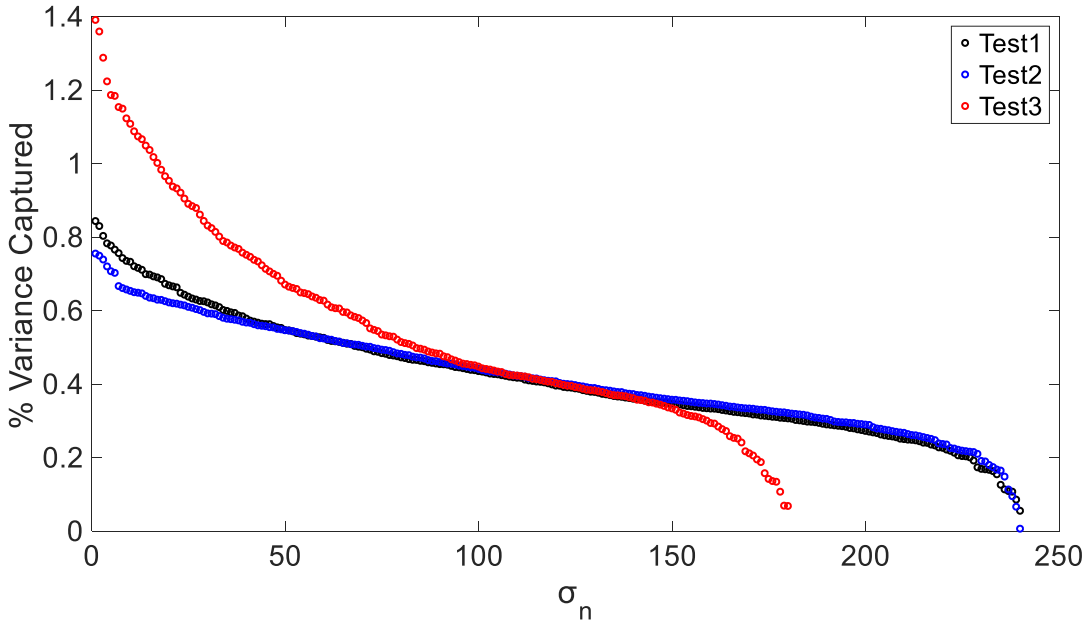

Figure 2 shows the variance captured by each σ_n for tests 1-3.

Dogg's "Gin and Juice" is shown in figure 1. The spectrograms were reshaped into row vectors and used to create the matrix X, which then underwent singular value decomposition (SVD). With the SVD of the spectrograms U, Σ, and V were analyzed and used for classification. The percent variance captured by each of the σ values for tests 1-3 are shown in figure 2. Unlike the first two reports the early σ_n do not capture a large portion of variance, with the first 20% (48 for tests 1 and 2, 36 for test 3) σ values capturing only 32%, 30%, and 36% percent, respectively. This shows that the dominant principal components do not do a good job of capturing the variance present across the spectrograms. From this point the matrix V is separated into artists/genres by row and used to create the training and test sets. A classification matrix assigns values 1-3 and was used to supervise the classification process. The predictive model is generated by the training and classification matrices, and tested using the test matrix. The results from the cross validation of the cross validation are shown as bar plots in figures 2-4 and table 1 shows the values for percentage correct predictions.

Comparisons of three different artist from three different genres resulted in the highest prediction percentage, and the results are shown in figure 3. This is the simplest case because it maximizes the out of class variance (between genres) and minimizes the in class variance (one artist's songs). Results from this test show that overall the prediction is correct 80% of the time. Between artists there is a large discrepancy in the percentage correctly identified. Snoop Dogg had the highest correct with 97% and the Beatles had the lowest with 60%.

Table 1: Cross Validation Results

Test #	Average % Correct	Artist/Genre 1 % Correct	Artist/Genre 2 % Correct	Artist/Genre 3 % Correct
1	79.67	82.13	97.38	59.50
2	50	31	89.38	30.25
3	74.06	82.83	88.17	51.17

Table 1 shows the percentage correctly identified from cross validation for the three artists/genres compared in each test and the average.

This difference is likely the result of Snoop Dogg's songs having less variety than the Beatles. The second test involved three artists in the same genre, which reduces the out of class variance and maintains similar class variance when compared to test 1.Figure 4 shows a bar plot of the percentage correctly identified. Results from test 2 were significantly worse, with the average correct identification being 49%. Comparing the artists the model did a poor job of predicting the Beatles and Jimi Hendrix songs, but identified the Rolling Stones correctly nearly 90% of the time. This shows how important the songs selected are for the prediction accuracy because if I had chosen someone other than Jimi Hendrix (that sounds significantly more different than the Beatles) the correct identification would increase for both, which was observed in test 1. The final test kept similar out of class variance compared to test 1, but increased the in class variance by using different artists from the same genre. Figure 5 shows a bar plot of the percentage correctly identified. The average percent correct for this test was 74%, which is similar to the results from test 1. This result is only about 6% lower than test 1, so the model is able to identify genres almost as well as specific artist. Rock lowered the average significantly because the correct identification was about 30% lower than rap or pop. This is expected from test 2 because we see how much in class variance is present in the samples chosen for the rock genre. The results from the three test match qualitatively with what was expected based on the musical variety in each of the X matrices.

Sec. V. Summary & Conclusions

Music artist and genre classification was attempted using a Naïve Bayes classifier. Three different tests were performed. The first compared three artists in different genres, the second three artists in the same genre, and the third multiple artists across three different genres. Spectrograms of five second samples of songs were taken and reformatted into a matrix X that underwent singular value decomposition. The resultant V matrix contained information about how the different samples project onto the principal component space (U), and this information was used with a Naïve Bayes classifier to generate the predictive model. Cross validation results are shown in table A and figures C-E. The model classified the artists and genres in tests 1 and 3 well, but struggled with different rock artists in test 2. These results are expected based on the amount of in class and out of class variance present in each data set.

Figure 3: Percent Correct of Cross Validation for Test 1

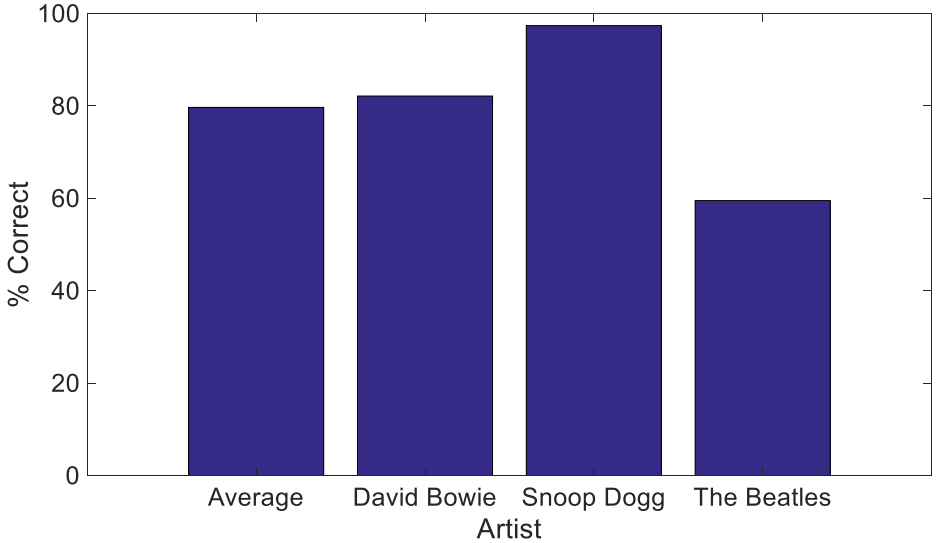

Figure 3 shows the percent correct for the three artists and the average for test 1.

Figure 4: Percent Correct of Cross Validation for Test 2

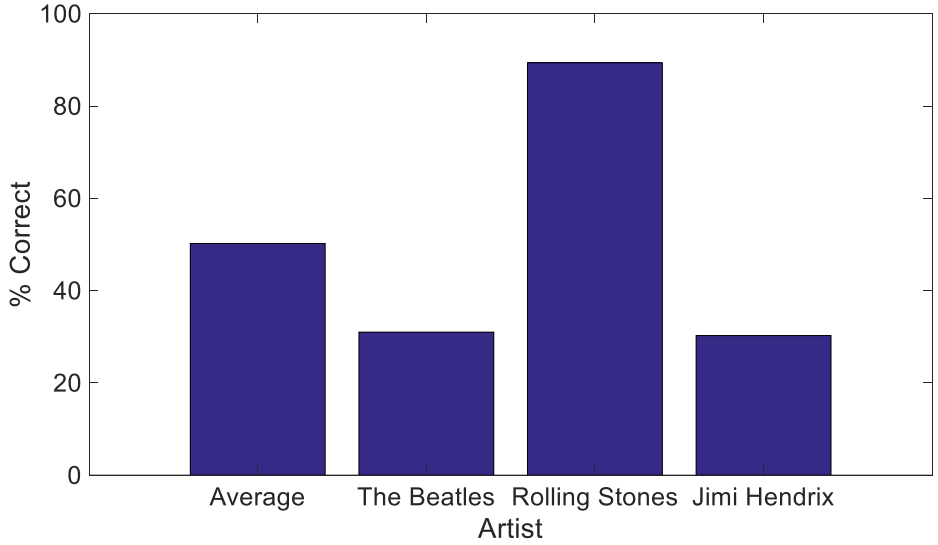

Figure 4 shows the percent correct for the three artists and the average for test 2.

Figure 5: Percent Correct of Cross Validation for Test 3

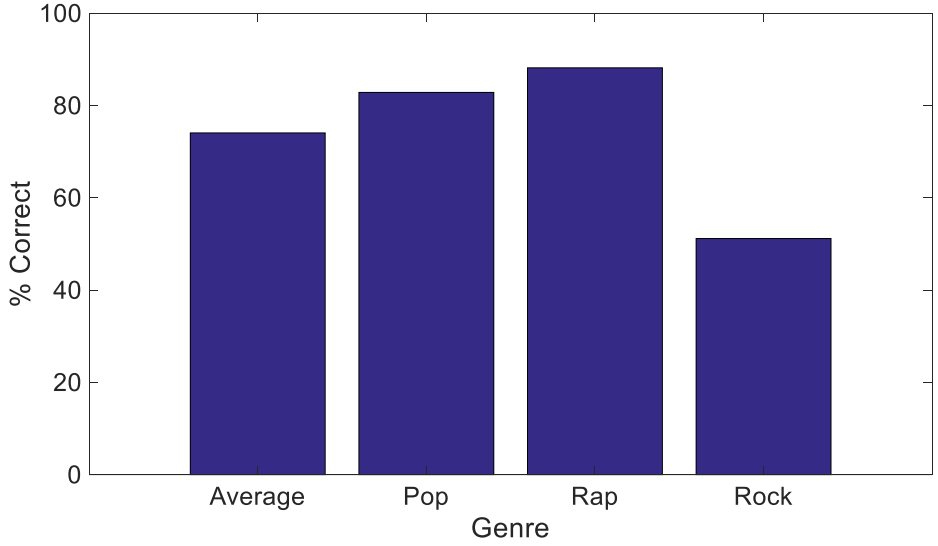

Figure 5 shows the percent correct for the three genres and the average for test 3.

References

1. Kutz, J. N. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*; Oxford University Press, Inc.: New York, NY, USA, **2013**.

Appendix A

MusicConvert(file, time) – Inputs an mp3 file and outputs a spectrogram of a five second segment as a row vector.

spectrogram(x) – Converts an mp3 file to a spectrogram.
Fitcnb(train, classify) – Trains a Naïve Bayes classifier model.

Appendix B

```
%% Import all of the song files

clear; clc;

t = 60;
mj_sample = cell(1,80);
sd_sample = cell(1,80);
tb_sample = cell(1,80);
rs_sample = cell(1,80);
jh_sample = cell(1,80);
bowie_sample = cell(1,80);
rock_sample = cell(1,60);
rap_sample = cell(1,60);
pop_sample = cell(1,60);

% Michael Jackson
myFolder = 'C:\Users\Brendon\Desktop\AMATH Music\Michael Jackson';
filePattern = fullfile(myFolder, '*.mp3');
theFiles = dir(filePattern);
for i = 1 : 80
  baseFileName = theFiles(i).name;
  fullFileName = fullfile(myFolder, baseFileName);
  mj_sample{i} = MusicConvert(fullFileName, t);
end


% Snoop Dogg
myFolder = 'C:\Users\Brendon\Desktop\AMATH Music\Snoop Dogg';
filePattern = fullfile(myFolder, '*.mp3');
theFiles = dir(filePattern);
for i = 1 : 80
  baseFileName = theFiles(i).name;
  fullFileName = fullfile(myFolder, baseFileName);
  sd_sample{i} = MusicConvert(fullFileName, t);
end

% The Beatles
myFolder = 'C:\Users\Brendon\Desktop\AMATH Music\The Beatles';
filePattern = fullfile(myFolder, '*.mp3');
theFiles = dir(filePattern);
```

```
for i = 1 : 80
  baseFileName = theFiles(i).name;
  fullFileName = fullfile(myFolder, baseFileName);
  tb_sample{i} = MusicConvert(fullFileName, t);
end

% Rolling Stones
myFolder = 'C:\Users\Brendon\Desktop\AMATH Music\Rolling Stones';
filePattern = fullfile(myFolder, '*.mp3');
theFiles = dir(filePattern);
for i = 1 : 80
  baseFileName = theFiles(i).name;
  fullFileName = fullfile(myFolder, baseFileName);
  rs_sample{i} = MusicConvert(fullFileName, t);
end

% Jimi Hendrix
myFolder = 'C:\Users\Brendon\Desktop\AMATH Music\Jimi Hendrix';
filePattern = fullfile(myFolder, '*.mp3');
theFiles = dir(filePattern);
for i = 1 : 80
  baseFileName = theFiles(i).name;
  fullFileName = fullfile(myFolder, baseFileName);
  jh_sample{i} = MusicConvert(fullFileName, t);
end

% David Bowie
myFolder = 'C:\Users\Brendon\Desktop\AMATH Music\David Bowie';
filePattern = fullfile(myFolder, '*.mp3');
theFiles = dir(filePattern);
for i = 1 : 80
  baseFileName = theFiles(i).name;
  fullFileName = fullfile(myFolder, baseFileName);
  bowie_sample{i} = MusicConvert(fullFileName,t);
end

% Rock
myFolder = 'C:\Users\Brendon\Desktop\AMATH Music\Rock';
filePattern = fullfile(myFolder, '*.mp3');
theFiles = dir(filePattern);
for i = 1 : 60
  baseFileName = theFiles(i).name;
  fullFileName = fullfile(myFolder, baseFileName);
  rock_sample{i} = MusicConvert(fullFileName, t);
end

% Rap
myFolder = 'C:\Users\Brendon\Desktop\AMATH Music\Rap';
```

```
filePattern = fullfile(myFolder, '*.mp3');
theFiles = dir(filePattern);
for i = 1 : 60
  baseFileName = theFiles(i).name;
  fullFileName = fullfile(myFolder, baseFileName);
  rap_sample{i} = MusicConvert(fullFileName, t);
end

% Pop
myFolder = 'C:\Users\Brendon\Desktop\AMATH Music\Pop';
filePattern = fullfile(myFolder, '*.mp3');
    theFiles = dir(filePattern);
    for i = 1 : 60
      baseFileName = theFiles(i).name;
      fullFileName = fullfile(myFolder, baseFileName);
      pop_sample{i} = MusicConvert(fullFileName, t);
    end
    %% Test 1

    n1 = length(bowie_sample);
    X1 = zeros(3*n1,294921);

    for i = 1:n1
       X1(i,:) = bowie_sample{i}(:,1);
       X1(i+(n1),:) = sd_sample{i}(:,1);
       X1(i+(2*n1),:) = tb_sample{i}(:,1);
    end

    [u1,s1,v1] = svd(X1','econ');

    x_bowie = v1(1:(n1),1:48);
    x_sd = v1((n1+1):(2*n1),1:48);
    x_tb = v1((2*n1+1):(3*n1),1:48);

    correct_1 = 0;
    correct_bowie = 0;
    correct_sd = 0;
    correct_tb = 0;
    incorrect_1 = 0;
    for q=1:50

    q1 = randperm(n1);
    q2 = randperm(n1);
    q3 = randperm(n1);

    n80_1 = round(n1*0.8);

    xtrain1 = [x_bowie(q1(1:n80_1),:); x_sd(q2(1:n80_1),:); x_tb(q3(1:n80_1),:)];
```

```
137.    xtest1 = [x_bowie(q1(n80_1+1:end),:); x_sd(q2(n80_1+1:end),:); x_tb(q3(n80_1+1:end),:)];
138.
139.    ctrain1 = [ones(n80_1,1); 2*ones(n80_1,1); 3*ones(n80_1,1)];
140.    verify1 = [ones((n1-n80_1),1); 2*ones((n1-n80_1),1); 3*ones((n1-n80_1),1)];
141.
142.    nb1 = fitcnb(xtrain1,ctrain1);
143.    predict1 = nb1.predict(xtest1);
144.
145.    for w = 1:(3*(n1-n80_1))
146.       if predict1(w)-verify1(w) == 0
147.          if w <= (n1-n80_1)
148.             correct_bowie = correct_bowie + 1;
149.          elseif w <= (2*(n1-n80_1))
150.             correct_sd = correct_sd + 1;
151.          else
152.             correct_tb = correct_tb + 1;
153.          end
154.          correct_1 = correct_1 + 1;
155.       else
156.          incorrect_1 = incorrect_1 + 1;
157.       end
158.    end
159.    end
160.
161.    correct_matrix1(1) = correct_1/(incorrect_1+correct_1)*100;
162.    correct_matrix1(2) = correct_bowie/((incorrect_1+correct_1)/3)*100;
163.    correct_matrix1(3) = correct_sd/((incorrect_1+correct_1)/3)*100;
164.    correct_matrix1(4) = correct_tb/((incorrect_1+correct_1)/3)*100;
165.    figure(10); bar(correct_matrix1); axis([0 5 0 100]); xlabel('Artist','FontSize',35); ylabel('%
    Correct','FontSize',35);
166.
167.    %% Test 2
168.
169.    n2 = length(tb_sample);
170.    X2 = zeros(n2,294921);
171.
172.    for i = 1:(n2)
173.       X2(i,:) = tb_sample{i}(:,1);
174.       X2(i+(n2),:) = rs_sample{i}(:,1);
175.       X2(i+(2*n2),:) = jh_sample{i}(:,1);
176.    end
177.
178.    [u2,s2,v2] = svd(X2','econ');
179.
180.    x_tb = v2(1:(n2),1:48);
181.    x_rs = v2((n2+1):(2*n2),1:48);
182.    x_jh = v2((2*n2+1):(3*n2),1:48);
183.
```

```
184.    correct_2 = 0;
185.    correct_tb = 0;
186.    correct_rs = 0;
187.    correct_jh = 0;
188.    incorrect_2 = 0;
189.
190.    for q=1:50
191.
192.    q1 = randperm(n2);
193.    q2 = randperm(n2);
194.    q3 = randperm(n2);
195.
196.    n80_2 = round(n2*0.8);
197.
198.    xtrain2 = [x_tb(q1(1:n80_2),:); x_rs(q2(1:n80_2),:); x_jh(q3(1:n80_2),:)];
199.    xtest2 = [x_tb(q1(n80_2+1:end),:); x_rs(q2(n80_2+1:end),:); x_jh(q3(n80_2+1:end),:)];
200.
201.    ctrain2 = [ones(n80_2,1); 2*ones(n80_2,1); 3*ones(n80_2,1)];
202.    verify2 = [ones((n2-n80_2),1); 2*ones((n2-n80_2),1); 3*ones((n2-n80_2),1)];
203.
204.    nb2 = fitcnb(xtrain2,ctrain2);
205.    predict1 = nb2.predict(xtest2);
206.
207.    for w = 1:(3*(n2-n80_2))
208.       if predict1(w)-verify2(w) == 0
209.          if w <= (n2-n80_2)
210.             correct_tb = correct_tb + 1;
211.          elseif w <= (2*(n2-n80_2))
212.             correct_rs = correct_rs + 1;
213.          else
214.             correct_jh = correct_jh + 1;
215.          end
216.          correct_2 = correct_2 + 1;
217.       else
218.          incorrect_2 = incorrect_2 + 1;
219.       end
220.    end
221.    end
222.
223.    correct_matrix2(1) = correct_2/(incorrect_2+correct_2)*100;
224.    correct_matrix2(2) = correct_tb/((incorrect_2+correct_2)/3)*100;
225.    correct_matrix2(3) = correct_rs/((incorrect_2+correct_2)/3)*100;
226.    correct_matrix2(4) = correct_jh/((incorrect_2+correct_2)/3)*100;
227.    figure(2); bar(correct_matrix2); axis([0 5 0 100]); xlabel('Artist','FontSize',35); ylabel('%
    Correct','FontSize',35);
228.
229.    %% Test 3
230.
```

```
n3 = length(rap_sample);
X3 = zeros(n3,294921);

for i = 1:(n3)
   X3(i,:) = pop_sample{i}(:,1);
   X3(i+(n3),:) = rap_sample{i}(:,1);
   X3(i+(2*n3),:) = rock_sample{i}(:,1);
end

[u3,s3,v3] = svd(X3','econ');

x_pop = v3(1:(n3),1:36);
x_rap = v3((n3+1):(2*n3),1:36);
x_rock = v3((2*n3+1):(3*n3),1:36);

correct_3 = 0;
correct_pop = 0;
correct_rap = 0;
correct_rock = 0;
incorrect_3 = 0;

for q=1:50

q1 = randperm(n3);
q2 = randperm(n3);
q3 = randperm(n3);

n80_3 = round(n3*0.8);

xtrain2 = [x_pop(q1(1:n80_3),:); x_rap(q2(1:n80_3),:); x_rock(q3(1:n80_3),:)];
xtest2 = [x_pop(q1(n80_3+1:end),:); x_rap(q2(n80_3+1:end),:); x_rock(q3(n80_3+1:end),:)];

ctrain2 = [ones(n80_3,1); 2*ones(n80_3,1); 3*ones(n80_3,1)];
verify2 = [ones((n3-n80_3),1); 2*ones((n3-n80_3),1); 3*ones((n3-n80_3),1)];

nb2 = fitcnb(xtrain2,ctrain2);
predict1 = nb2.predict(xtest2);

for w = 1:(3*(n3-n80_3))
   if predict1(w)-verify2(w) == 0
      if w <= (n3-n80_3)
         correct_pop = correct_pop + 1;
      elseif w <= (2*(n3-n80_3))
         correct_rap = correct_rap + 1;
      else
         correct_rock = correct_rock + 1;
      end
      correct_3 = correct_3 + 1;
```

```
279.        else
280.            incorrect_3 = incorrect_3 + 1;
281.        end
282.    end
283.    end
284.
285.    correct_matrix3(1) = correct_3/(incorrect_3+correct_3)*100;
286.    correct_matrix3(2) = correct_pop/((incorrect_3+correct_3)/3)*100;
287.    correct_matrix3(3) = correct_rap/((incorrect_3+correct_3)/3)*100;
288.    correct_matrix3(4) = correct_rock/((incorrect_3+correct_3)/3)*100;
289.    figure(3); bar(correct_matrix3); axis([0 5 0 100]); xlabel('Genre','FontSize',35); ylabel('%
    Correct','FontSize',35);
290.
291.    %%
292.
293.    ds1 = diag(s1);
294.    ds2 = diag(s2);
295.    ds3 = diag(s3);
296.    figure(4); plot(ds1./sum(ds1)*100,'ko','LineWidth',[2]); hold on;
297.    plot(ds2./sum(ds2)*100,'bo','LineWidth',[2]);
298.    plot(ds3./sum(ds3)*100,'ro','LineWidth',[2]);
299.    xlabel('?_n','FontSize',35); ylabel('% Variance Captured','FontSize',35);
300.
301.    function [song_reshape, mono] = MusicConvert(filename, t)
302.        [stereo,fs] = audioread(filename);
303.        mono = (stereo(:,1)+ stereo(:,2))/2;
304.        t1 = t*fs;
305.        t2 = (t+5)*fs;
306.        song_crop = mono(t1:t2);
307.        song_sample = spectrogram(song_crop,fs);
308.        song_sample_real = real(song_sample);
309.        song_reshape = reshape(song_sample_real,[294921,1]);
310.    end
```