```
%% TEST 1
clc; clear all; close all;
%% Import MP3 Files
% Choice: Michael Jackson, Soundgarden, Beethoven
% myDir = 'C:\Users\Johnnia\Desktop\46\UW\SchoolWorks\2018Winter\AMATH482\HW3\MP3';
% [mj, Rmj] = importData([myDir '\MichaelJackson']);
% [sg, Rsg] = importData([myDir '\SoundGarden']);
% [bee, Rbee] = importData([myDir '\Beethoven']);
%
% % Resample the data
% mj = resample(mj, 1, 2);
% sg = resample(sg, 1, 2);
% bee = resample(bee, 1, 2);
%
% % Save for further use
% save('mj.mat', 'mj');
% save('sg.mat', 'sg');
% save('bee.mat', 'bee');

% Load if possible
load('mj.mat');
load('sg.mat');
load('bee.mat');

% Make into one data matrix
X = [mj, sg, bee];

%% Label

for i = 1:size(mj, 2)
    labs{i} = 'MichaelJackson';
end

for i = 1:size(sg, 2)
    labs{i+size(mj, 2)} = 'SoundGarden';
end

for i = 1:size(bee,2)
    labs{i+size(mj, 2)+size(sg, 2)} = 'Beethoven';
end

%% Get Distinguishable Features

zcd = dsp.ZeroCrossingDetector;
for i = 1:size(X,2)

    % Count the number of times the signal crosses zero.
    cross0(i,:) = double(zcd(X(:,i)));

    % Get the max, min, mean, variance and norm of the signal
    maxV(i,:) = max(abs(X(:,i)));
    minV(i,:) = min(abs(X(:,i)));
    meanV(i,:) = mean(X(:,i));
    varV(i,:) = var(X(:,i));
    normV(i,:) = norm(X(:,i));

    % Get the first 2 highest frequencies from the FFT.
    fs = abs(fft(X(:,i)));
    fs = fs(1:floor(end/2));
    [~,f1(i,:)] = max(fs);
    fs(f1(i)) = 0;
    [~,f2(i,:)] = max(fs);
end

% Create the feature matrix
```

```
featureData = [cross0, maxV, minV, meanV, varV, normV, f1, f2];

%% Train the classifier and get the test result for 5 loops
acc_dt = zeros(5, 1);
acc_nb = zeros(5, 1);
for i = 1:5
    acc_dt(i) = Classifier(featureData, labs, 'DecisionTree');
    acc_nb(i) = Classifier(featureData, labs, 'NaiveBayes');
end

%% TEST 2
clc; clear all; close all;
%% Import MP3 Files
% % Choice: Alice In Chains, Soundgarden, Pearl Jam
% myDir = 'C:\Users\Johnnia\Desktop\46\UW\SchoolWorks\2018Winter\AMATH482\HW3\MP3';
% [aic, Raic] = importData([myDir '\AliceInChains']);
% [sg, Rsg] = importData([myDir '\SoundGarden']);
% [pj, Rpj] = importData([myDir '\PearlJam']);
%
% % Resample the data
% aic = resample(aic, 1, 2);
% sg = resample(sg, 1, 2);
% pj = resample(pj, 1, 2);
%
% % Save for further use
% save('aic.mat', 'aic');
% save('sg.mat', 'sg');
% save('pj.mat', 'pj');

% Load if possible
load('aic.mat');
load('sg.mat');
load('pj.mat');

% Make into one data matrix
X = [aic, sg, pj];

%% Label

for i = 1:size(aic, 2)
    labs{i} = 'AliceInChains';
end

for i = 1:size(sg, 2)
    labs{i+size(aic, 2)} = 'SoundGarden';
end

for i = 1:size(pj,2)
    labs{i+size(aic, 2)+size(sg, 2)} = 'PearlJam';
end

%% Get Distinguishable Features

zcd = dsp.ZeroCrossingDetector;
for i = 1:size(X,2)

    % Count the number of times the signal crosses zero.
    cross0(i,:) = double(zcd(X(:,i)));

    % Get the max, min, mean, variance and norm of the signal
    maxV(i,:) = max(abs(X(:,i)));
    minV(i,:) = min(abs(X(:,i)));
    meanV(i,:) = mean(X(:,i));
    varV(i,:) = var(X(:,i));
    normV(i,:) = norm(X(:,i));
```

```
131
132         % Get the first 2 highest frequencies from the FFT.
133         fs = abs(fft(X(:,i)));
134         fs = fs(1:floor(end/2));
135         [~,f1(i,:)] = max(fs);
136         fs(f1(i)) = 0;
137         [~,f2(i,:)] = max(fs);
138
139         wavl(i,:) = sum(abs(diff(X(:,i))));
140     end
141
142     % Create the feature matrix
143     featureData = [cross0, maxV, minV, meanV, varV, normV, f1, f2, wavl];
144
145     %% Train the classifier and get the test result for 5 loops
146     acc_dt = zeros(5, 1);
147     acc_nb = zeros(5, 1);
148     for i = 1:5
149         acc_dt(i) = Classifier(featureData, labs, 'DecisionTree');
150         acc_nb(i) = Classifier(featureData, labs, 'NaiveBayes');
151     end
152
153     %% TEST 3
154     clc; clear all; close all;
155     %% Import MP3 Files
156     % Choice: Alice In Chains, Soundgarden, Pearl Jam
157     myDir = 'C:\Users\Johnnia\Desktop\46\UW\SchoolWorks\2018Winter\AMATH482\HW3\genres';
158     [blues, Rb] = importData([myDir '\blues']);
159     [classical, Rclas] = importData([myDir '\classical']);
160     [country, Rcoun] = importData([myDir '\country']);
161     [disco, Rd] = importData([myDir '\disco']);
162     [hiphop, Rhh] = importData([myDir '\hiphop']);
163     [jazz, Rjz] = importData([myDir '\jazz']);
164     [metal, Rm] = importData([myDir '\metal']);
165     [pop, Rp] = importData([myDir '\pop']);
166     [reggae, Rr] = importData([myDir '\reggae']);
167     [rock, Rrc] = importData([myDir '\rock']);
168
169     %% Resample the data
170     blues = resample(blues, 1, 2);
171     classical = resample(classical, 1, 2);
172     country = resample(country, 1, 2);
173     disco = resample(disco, 1, 2);
174     hiphop = resample(hiphop, 1, 2);
175     hiphop = hiphop(:, 1:end-1);
176     jazz = resample(jazz, 1, 2);
177     metal = resample(metal, 1, 2);
178     pop = resample(pop, 1, 2);
179     reggae = resample(reggae, 1, 2);
180     rock = resample(rock, 1, 2);
181
182     % Save for further use
183     save('blues.mat', 'blues');
184     save('classical.mat', 'classical');
185     save('country.mat', 'country');
186     save('disco.mat', 'disco');
187     save('hiphop.mat', 'hiphop');
188     save('jazz.mat', 'jazz');
189     save('metal.mat', 'metal');
190     save('pop.mat', 'pop');
191     save('reggae.mat', 'reggae');
192     save('rock.mat', 'rock');
193
194     %% Load if possible
195     load('blues.mat');
```

```
load('classical.mat');
load('country.mat');
load('disco.mat');
load('hiphop.mat');
load('jazz.mat');
load('metal.mat');
load('pop.mat');
load('reggae.mat');
load('rock.mat');

% Make into one data matrix
X = [blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock];

%% Label

for i = 1:size(blues, 2)
    labs{i} = 'blues';
end

for i = 1:size(classical, 2)
    labs{i+size(blues, 2)} = 'classical';
end

for i = 1:size(country,2)
    labs{i+size(blues, 2)+size(classical, 2)} = 'country';
end

for i = 1:size(disco,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)} = 'disco';
end

for i = 1:size(hiphop,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)} = 'hiphop';
end

for i = 1:size(jazz,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)+size(hiphop, 2)} = 'jazz';
end

for i = 1:size(metal,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)} = 'metal';
end

for i = 1:size(pop,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)...
        +size(metal, 2)} = 'pop';
end

for i = 1:size(reggae,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)...
        +size(metal, 2)+size(pop, 2)} = 'reggae';
end

for i = 1:size(rock,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)...
        +size(metal, 2)+size(pop, 2)+size(reggae, 2)} = 'rock';
end

%% Label2
```

```
for i = 1:size(rock, 2)
    labs{i} = 'rock';
end

for i = 1:size(jazz, 2)
    labs{i+size(rock, 2)} = 'jazz';
end

for i = 1:size(classical,2)
    labs{i+size(rock, 2)+size(jazz, 2)} = 'classical';
end
%% Get Distinguishable Features

zcd = dsp.ZeroCrossingDetector;
for i = 1:size(X,2)

    % Count the number of times the signal crosses zero.
    cross0(i,:) = double(zcd(X(:,i)));

    % Get the max, min, mean, variance and norm of the signal
    maxV(i,:) = max(abs(X(:,i)));
    minV(i,:) = min(abs(X(:,i)));
    meanV(i,:) = mean(X(:,i));
    varV(i,:) = var(X(:,i));
    normV(i,:) = norm(X(:,i));

    % Get the first 2 highest frequencies from the FFT.
    fs = abs(fft(X(:,i)));
    fs = fs(1:floor(end/2));
    [~,f1(i,:)] = max(fs);
    fs(f1(i)) = 0;
    [~,f2(i,:)] = max(fs);

    wavl(i,:) = sum(abs(diff(X(:,i))));
end

% Create the feature matrix
featureData = [cross0, maxV, minV, meanV, varV, normV, f1, f2, wavl];

%% Train the classifier and get the test result for 5 loops
acc_dt = zeros(5, 1);
acc_nb = zeros(5, 1);
for i = 1:5
    acc_dt(i) = Classifier(featureData, labs, 'DecisionTree');
    acc_nb(i) = Classifier(featureData, labs, 'NaiveBayes');
end


%%
clc; clear all; close all;
%% Comparison
load('dt_1.mat'); load('nb_1.mat');
dt1 = acc_dt; nb1 = acc_nb;
load('dt_2.mat'); load('nb_2.mat');
dt2 = acc_dt; nb2 = acc_nb;
load('dt_3.mat'); load('nb_3.mat');
dt3 = acc_dt; nb3 = acc_nb;

dt1_mean = mean(dt1); dt1_var = var(dt1);
nb1_mean = mean(nb1); nb1_var = var(nb1);
dt2_mean = mean(dt2); dt2_var = var(dt2);
nb2_mean = mean(nb2); nb2_var = var(nb2);
dt3_mean = mean(dt3); dt3_var = var(dt3);
nb3_mean = mean(nb3); nb3_var = var(nb3);

```

```
dtmeans = [dt1_mean, dt2_mean, dt3_mean];
nbmeans = [nb1_mean, nb2_mean, nb3_mean];
dtvars = [dt1_var, dt2_var, dt3_var];
nbvars = [nb1_var, nb2_var, nb3_var];

figure
hb = bar([1, 2, 3], [dtmeans, nbmeans]);
set(hb(1), 'FaceColor','r')
set(hb(2), 'FaceColor','b')
set(hb(3), 'FaceColor','g')

%%
data = [[dt1_mean, nb1_mean]; [dt2_mean, nb2_mean]; [dt3_mean, nb3_mean]];
figure(1)
b = bar(data);
err = [[dt1_var, nb1_var]; [dt2_var, nb2_var]; [dt3_var, nb3_var]];
hold on;
title('Comparison Between Different Tests and Classifiers')
xlabel('Test #');
ylabel('Accuracy');
legend('Decision Tree', 'Naive Bayes');
grid on;

function accuracy = Classifier(dat,labs, classifier)
n = size(dat,1);
trainSize = floor(n*0.7);
testSize = floor(n*0.3);

%% Divide data to be training set and testing set
randIndex = randperm(n);
randData = dat(randIndex,:);
randlabs = labs(randIndex);

% First 70% to be training data
train = randData(1:trainSize,:);
trainlabs = randlabs(1:trainSize);

% The other 30% to be testing data
test = randData(trainSize+1:end,:);
testlabs = randlabs(trainSize+1:end);

%% Perform SVD

[u,s,v] = svd(train','econ');

% Compute energy
energy = diag(s)/sum(diag(s));

% Find modes with less than 10% energy
i = find(energy < 0.1);

% If the above all modes are within 10%
if isempty(i)
    endInd = size(v,2);
else
    endInd = i(1)-1;
end

% Get the data with only modes with more than 10% energy
train = v(:,1:endInd);
test = s\u'*test';
test = test(1:endInd,:)';

%% Train
if (strcmp(classifier, 'DecisionTree'))
```

```
    model = TreeBagger(30, train, trainlabs);
elseif (strcmp(classifier, 'NaiveBayes'))
    model = fitcnb(train, trainlabs);
end
%% Test

predictions{testSize,1} = [];
correctness = zeros(testSize, 1);

for i = 1:testSize
    predictions{i,:} = predict(model, test(i,:));
    correctness(i) = strcmp(predictions{i,:}, testlabs{i});
end

accuracy = 100*sum(correctness)/length(correctness);

end

function [dat, Fs] = importData(myDir)

musics = dir(fullfile(myDir));
musics = musics(3:end, :);
n = length(musics);

dat = [];

for i = 1:n
    [song, Fs] = audioread(fullfile(myDir, musics(i).name));
    vectorSong = song(:,1);
    dat = [dat; vectorSong];
end

% Get the number of frames in 5 seconds
nIn5 = Fs*5;

% Get the number of 5-second clips
nClips = floor(length(dat)/nIn5);

% Trim the data matrix
dat = dat(1:nIn5*nClips);

% Reshape the data matrix
dat = reshape(dat,[nIn5, nClips]);

end
```