

```

1  %% TEST 1
2  clc; clear all; close all;
3  %% Import MP3 Files
4  % Choice: Michael Jackson, Soundgarden, Beethoven
5  % myDir = 'C:\Users\Johnnia\Desktop\46\UW\SchoolWorks\2018Winter\AMATH482\HW3\MP3';
6  % [mj, Rmj] = importData([myDir '\MichaelJackson']);
7  % [sg, Rsg] = importData([myDir '\SoundGarden']);
8  % [bee, Rbee] = importData([myDir '\Beethoven']);
9  %
10 % % Resample the data
11 % mj = resample(mj, 1, 2);
12 % sg = resample(sg, 1, 2);
13 % bee = resample(bee, 1, 2);
14 %
15 % % Save for further use
16 % save('mj.mat', 'mj');
17 % save('sg.mat', 'sg');
18 % save('bee.mat', 'bee');
19
20 % Load if possible
21 load('mj.mat');
22 load('sg.mat');
23 load('bee.mat');
24
25 % Make into one data matrix
26 X = [mj, sg, bee];
27
28 %% Label
29
30 for i = 1:size(mj, 2)
31     labs{i} = 'MichaelJackson';
32 end
33
34 for i = 1:size(sg, 2)
35     labs{i+size(mj, 2)} = 'SoundGarden';
36 end
37
38 for i = 1:size(bee,2)
39     labs{i+size(mj, 2)+size(sg, 2)} = 'Beethoven';
40 end
41
42 %% Get Distinguishable Features
43
44 zcd = dsp.ZeroCrossingDetector;
45 for i = 1:size(X,2)
46
47     % Count the number of times the signal crosses zero.
48     cross0(i,:) = double(zcd(X(:,i)));
49
50     % Get the max, min, mean, variance and norm of the signal
51     maxV(i,:) = max(abs(X(:,i)));
52     minV(i,:) = min(abs(X(:,i)));
53     meanV(i,:) = mean(X(:,i));
54     varV(i,:) = var(X(:,i));
55     normV(i,:) = norm(X(:,i));
56
57     % Get the first 2 highest frequencies from the FFT.
58     fs = abs(fft(X(:,i)));
59     fs = fs(1:floor(end/2));
60     [~,f1(i,:)] = max(fs);
61     fs(f1(i)) = 0;
62     [~,f2(i,:)] = max(fs);
63 end
64
65 % Create the feature matrix

```

```

66 featureData = [cross0, maxV, minV, meanV, varV, normV, f1, f2];
67
68 %% Train the classifier and get the test result for 5 Loops
69 acc_dt = zeros(5, 1);
70 acc_nb = zeros(5, 1);
71 for i = 1:5
72     acc_dt(i) = Classifier(featureData, labs, 'DecisionTree');
73     acc_nb(i) = Classifier(featureData, labs, 'NaiveBayes');
74 end
75
76 %% TEST 2
77 clc; clear all; close all;
78 %% Import MP3 Files
79 % % Choice: Alice In Chains, Soundgarden, Pearl Jam
80 % myDir = 'C:\Users\Johnnia\Desktop\46\UW\SchoolWorks\2018Winter\AMATH482\HW3\MP3';
81 % [aic, Raic] = importData([myDir '\AliceInChains']);
82 % [sg, Rsg] = importData([myDir '\SoundGarden']);
83 % [pj, Rpj] = importData([myDir '\PearlJam']);
84 %
85 % % Resample the data
86 % aic = resample(aic, 1, 2);
87 % sg = resample(sg, 1, 2);
88 % pj = resample(pj, 1, 2);
89 %
90 % % Save for further use
91 % save('aic.mat', 'aic');
92 % save('sg.mat', 'sg');
93 % save('pj.mat', 'pj');
94
95 % Load if possible
96 load('aic.mat');
97 load('sg.mat');
98 load('pj.mat');
99
100 % Make into one data matrix
101 X = [aic, sg, pj];
102
103 %% Label
104
105 for i = 1:size(aic, 2)
106     labs{i} = 'AliceInChains';
107 end
108
109 for i = 1:size(sg, 2)
110     labs{i+size(aic, 2)} = 'SoundGarden';
111 end
112
113 for i = 1:size(pj, 2)
114     labs{i+size(aic, 2)+size(sg, 2)} = 'PearlJam';
115 end
116
117 %% Get Distinguishable Features
118
119 zcd = dsp.ZeroCrossingDetector;
120 for i = 1:size(X, 2)
121
122     % Count the number of times the signal crosses zero.
123     cross0(i, :) = double(zcd(X(:, i)));
124
125     % Get the max, min, mean, variance and norm of the signal
126     maxV(i, :) = max(abs(X(:, i)));
127     minV(i, :) = min(abs(X(:, i)));
128     meanV(i, :) = mean(X(:, i));
129     varV(i, :) = var(X(:, i));
130     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');

```

```

196 load('classical.mat');
197 load('country.mat');
198 load('disco.mat');
199 load('hiphop.mat');
200 load('jazz.mat');
201 load('metal.mat');
202 load('pop.mat');
203 load('reggae.mat');
204 load('rock.mat');
205
206 % Make into one data matrix
207 X = [blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock];
208
209 %% Label
210
211 for i = 1:size(blues, 2)
212     labs{i} = 'blues';
213 end
214
215 for i = 1:size(classical, 2)
216     labs{i+size(blues, 2)} = 'classical';
217 end
218
219 for i = 1:size(country,2)
220     labs{i+size(blues, 2)+size(classical, 2)} = 'country';
221 end
222
223 for i = 1:size(disco,2)
224     labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)} = 'disco';
225 end
226
227 for i = 1:size(hiphop,2)
228     labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
229         +size(disco, 2)} = 'hiphop';
230 end
231
232 for i = 1:size(jazz,2)
233     labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
234         +size(disco, 2)+size(hiphop, 2)} = 'jazz';
235 end
236
237 for i = 1:size(metal,2)
238     labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
239         +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)} = 'metal';
240 end
241
242 for i = 1:size(pop,2)
243     labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
244         +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)...
245         +size(metal, 2)} = 'pop';
246 end
247
248 for i = 1:size(reggae,2)
249     labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
250         +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)...
251         +size(metal, 2)+size(pop, 2)} = 'reggae';
252 end
253
254 for i = 1:size(rock,2)
255     labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
256         +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)...
257         +size(metal, 2)+size(pop, 2)+size(reggae, 2)} = 'rock';
258 end
259
260 %% Label2

```

```

261 for i = 1:size(rock, 2)
262     labs{i} = 'rock';
263 end
264
265 for i = 1:size(jazz, 2)
266     labs{i+size(rock, 2)} = 'jazz';
267 end
268
269 for i = 1:size(classical,2)
270     labs{i+size(rock, 2)+size(jazz, 2)} = 'classical';
271 end
272 %% Get Distinguishable Features
273
274 zcd = dsp.ZeroCrossingDetector;
275 for i = 1:size(X,2)
276
277     % Count the number of times the signal crosses zero.
278     cross0(i,:) = double(zcd(X(:,i)));
279
280     % Get the max, min, mean, variance and norm of the signal
281     maxV(i,:) = max(abs(X(:,i)));
282     minV(i,:) = min(abs(X(:,i)));
283     meanV(i,:) = mean(X(:,i));
284     varV(i,:) = var(X(:,i));
285     normV(i,:) = norm(X(:,i));
286
287     % Get the first 2 highest frequencies from the FFT.
288     fs = abs(fft(X(:,i)));
289     fs = fs(1:floor(end/2));
290     [~,f1(i,:)] = max(fs);
291     fs(f1(i)) = 0;
292     [~,f2(i,:)] = max(fs);
293
294     wav1(i,:) = sum(abs(diff(X(:,i))));
295 end
296
297 % Create the feature matrix
298 featureData = [cross0, maxV, minV, meanV, varV, normV, f1, f2, wav1];
299
300 %% Train the classifier and get the test result for 5 Loops
301 acc_dt = zeros(5, 1);
302 acc_nb = zeros(5, 1);
303 for i = 1:5
304     acc_dt(i) = Classifier(featureData, labs, 'DecisionTree');
305     acc_nb(i) = Classifier(featureData, labs, 'NaiveBayes');
306 end
307
308
309 %%
310 clc; clear all; close all;
311 %% Comparison
312 load('dt_1.mat'); load('nb_1.mat');
313 dt1 = acc_dt; nb1 = acc_nb;
314 load('dt_2.mat'); load('nb_2.mat');
315 dt2 = acc_dt; nb2 = acc_nb;
316 load('dt_3.mat'); load('nb_3.mat');
317 dt3 = acc_dt; nb3 = acc_nb;
318
319 dt1_mean = mean(dt1); dt1_var = var(dt1);
320 nb1_mean = mean(nb1); nb1_var = var(nb1);
321 dt2_mean = mean(dt2); dt2_var = var(dt2);
322 nb2_mean = mean(nb2); nb2_var = var(nb2);
323 dt3_mean = mean(dt3); dt3_var = var(dt3);
324 nb3_mean = mean(nb3); nb3_var = var(nb3);
325

```

```

326 dtmeans = [dt1_mean, dt2_mean, dt3_mean];
327 nbmeans = [nb1_mean, nb2_mean, nb3_mean];
328 dtvars = [dt1_var, dt2_var, dt3_var];
329 nbvars = [nb1_var, nb2_var, nb3_var];
330
331 figure
332 hb = bar([1, 2, 3], [dtmeans, nbmeans]);
333 set(hb(1), 'FaceColor','r')
334 set(hb(2), 'FaceColor','b')
335 set(hb(3), 'FaceColor','g')
336
337 %%
338 data = [[dt1_mean, nb1_mean]; [dt2_mean, nb2_mean]; [dt3_mean, nb3_mean]];
339 figure(1)
340 b = bar(data);
341 err = [[dt1_var, nb1_var]; [dt2_var, nb2_var]; [dt3_var, nb3_var]];
342 hold on;
343 title('Comparison Between Different Tests and Classifiers')
344 xlabel('Test #');
345 ylabel('Accuracy');
346 legend('Decision Tree', 'Naive Bayes');
347 grid on;
348
349 function accuracy = Classifier(dat, labs, classifier)
350 n = size(dat,1);
351 trainSize = floor(n*0.7);
352 testSize = floor(n*0.3);
353
354 %% Divide data to be training set and testing set
355 randIndex = randperm(n);
356 randData = dat(randIndex,:);
357 randlabs = labs(randIndex);
358
359 % First 70% to be training data
360 train = randData(1:trainSize,:);
361 trainlabs = randlabs(1:trainSize);
362
363 % The other 30% to be testing data
364 test = randData(trainSize+1:end,:);
365 testlabs = randlabs(trainSize+1:end);
366
367 %% Perform SVD
368
369 [u,s,v] = svd(train','econ');
370
371 % Compute energy
372 energy = diag(s)/sum(diag(s));
373
374 % Find modes with less than 10% energy
375 i = find(energy < 0.1);
376
377 % If the above all modes are within 10%
378 if isempty(i)
379     endInd = size(v,2);
380 else
381     endInd = i(1)-1;
382 end
383
384 % Get the data with only modes with more than 10% energy
385 train = v(:,1:endInd);
386 test = s\ u'*test';
387 test = test(1:endInd,:);
388
389 %% Train
390 if (strcmp(classifier, 'DecisionTree'))

```

```

391     model = TreeBagger(30, train, trainlabs);
392 elseif (strcmp(classifier, 'NaiveBayes'))
393     model = fitcnb(train, trainlabs);
394 end
395 %% Test
396
397 predictions{testSize,1} = [];
398 correctness = zeros(testSize, 1);
399
400 for i = 1:testSize
401     predictions{i,:} = predict(model, test(i,:));
402     correctness(i) = strcmp(predictions{i,:), testlabs{i});
403 end
404
405 accuracy = 100*sum(correctness)/length(correctness);
406
407 end
408
409 function [dat, Fs] = importData(myDir)
410
411 musics = dir(fullfile(myDir));
412 musics = musics(3:end, :);
413 n = length(musics);
414
415 dat = [];
416
417 for i = 1:n
418     [song, Fs] = audioread(fullfile(myDir, musics(i).name));
419     vectorSong = song(:,1);
420     dat = [dat; vectorSong];
421 end
422
423 % Get the number of frames in 5 seconds
424 nIn5 = Fs*5;
425
426 % Get the number of 5-second clips
427 nClips = floor(length(dat)/nIn5);
428
429 % Trim the data matrix
430 dat = dat(1:nIn5*nClips);
431
432 % Reshape the data matrix
433 dat = reshape(dat,[nIn5, nClips]);
434
435 end

```