```
ads =
audioDatastore("C:\Users\ROG\Downloads\audios\proyecto\aumentos2","IncludeSubfolders
",true ,"FileExtensions",".wav","LabelSource","foldernames");
```

The splitEachLabel function of audioDatastore splits the datastore into two or more datastores. The resulting datastores have the specified proportion of the audio files from each label. In this example, the datastore is split into two parts. 80% of the data for each label is used for training, and the remaining 20% is used for testing. The countEachLabel function of audioDatastore is used to count the number of audio files per label. In this example, the label identifies the speaker.

```
[adsTrain,adsTest] = splitEachLabel(ads,0.8);
```

Display the datastore and the number of speakers in the train datastore.

```
adsTrain
```

```
adsTrain =
  audioDatastore with properties:
                       Files: {
                                ...\Downloads\audios\proyecto\aumentos2\adrian\augmented_10_adrian1.wav';
                                ...\Downloads\audios\proyecto\aumentos2\adrian\augmented_10_adrian10.wav';
                               ' ...\Downloads\audios\proyecto\aumentos2\adrian\augmented_10_adrian2.wav'
                               ... and 1373 more
                     Folders: {
                               C:\Users\ROG\Downloads\audios\proyecto\aumentos2'
                      Labels: [adrian; adrian; adrian ... and 1373 more categorical]
   AlternateFileSystemRoots: {}
              OutputDataType: 'double'
           OutputEnvironment: 'cpu'
                                        "flac"
      SupportedOutputFormats: ["wav"
                                                   "ogg"
                                                            "opus"
                                                                      "mp3"
                                                                                "mp4"
                                                                                         "m4a"]
         DefaultOutputFormat: "wav"
```

trainDatastoreCount = countEachLabel(adsTrain)

trainDatastoreCount = 17×2 table

	Label	Count
1	adrian	80
2	alfredo	80
3	andre	80
4	aris	80
5	bruno	80
6	dan	80
7	fredy	80
8	it	80
9	ivan	80

	Label	Count
10	juanpaulo	80
11	majo	80
12	mar	80
13	olguin	80
14	oscar	88
15	pao	80
16	yestli	80
17	yonathan	88

Display the datastore and the number of speakers in the test datastore.

```
adsTest
```

```
adsTest =
  audioDatastore with properties:
                       Files: {
                                ...\Downloads\audios\proyecto\aumentos2\adrian\augmented_8_adrian1.wav';
                              '...\Downloads\audios\proyecto\aumentos2\adrian\augmented_8_adrian10.wav';
                                ...\Downloads\audios\proyecto\aumentos2\adrian\augmented 8 adrian2.wav'
                               ... and 341 more
                     Folders: {
                              'C:\Users\ROG\Downloads\audios\proyecto\aumentos2'
                      Labels: [adrian; adrian; adrian ... and 341 more categorical]
    AlternateFileSystemRoots: {}
              OutputDataType: 'double'
           OutputEnvironment: 'cpu'
      SupportedOutputFormats: ["wav"
                                        "flac"
                                                   "ogg"
                                                            "opus"
                                                                      "mp3"
                                                                               "mp4"
                                                                                        "m4a"]
         DefaultOutputFormat: "wav"
```

testDatastoreCount = countEachLabel(adsTest)

test[DatastoreCoun	t =	17×2	tabl
	Label		Coun	t
1	adrian			20
2	alfredo			20
3	andre			20
4	aris			20
5	bruno			20
6	dan			20
7	fredy			20
8	it			20
9	ivan			20

Label Count 10 juanpaulo 20 11 majo 20 12 mar 20 13 olguin 20 14 oscar 22			
11 majo 20 12 mar 20 13 olguin 20		Label	Count
12 mar 20 13 olguin 20	10	juanpaulo	20
13 olguin 20	11	majo	20
olgulii 20	12	mar	20
14 oscar 22	13	olguin	20
	14	oscar	22
15 pao 20	15	pao	20
16 yestli 20	16	yestli	20
yonathan 22	17	yonathan	22

To preview the content of your datastore, read a sample file and play it using your default audio device.

```
[sampleTrain,dsInfo] = read(adsTrain);
sound(sampleTrain,dsInfo.SampleRate)
```

Reading from the train datastore pushes the read pointer so that you can iterate through the database. Reset the train datastore to return the read pointer to the start for the following feature extraction.

```
reset(adsTrain)
```

Feature Extraction

Extract pitch and MFCC features from each frame that corresponds to voiced speech in the training datastore. Audio Toolbox™ provides audioFeatureExtractor so that you can quickly and efficiently extract multiple features. Configure an audioFeatureExtractor to extract pitch, short-time energy, zcr, and MFCC.

```
fs = dsInfo.SampleRate;
windowLength = round(0.03*fs);
overlapLength = round(0.025*fs);
afe = audioFeatureExtractor(SampleRate=fs, ...
    Window=hamming(windowLength, "periodic"), OverlapLength=overlapLength, ...
    zerocrossrate=true, shortTimeEnergy=true, pitch=true, mfcc=true);
```

When you call the extract function of audioFeatureExtractor, all features are concatenated and returned in a matrix. You can use the info function to determine which columns of the matrix correspond to which features.

Extract features from the data set.

```
features = [];
labels = [];
energyThreshold = 0.015;
zcrThreshold = 0.5;
allFeatures = extract(afe,adsTrain);
allLabels = adsTrain.Labels;
for ii = 1:numel(allFeatures)
    thisFeature = allFeatures{ii};
    isSpeech = thisFeature(:,featureMap.shortTimeEnergy) > energyThreshold;
    isVoiced = thisFeature(:,featureMap.zerocrossrate) < zcrThreshold;</pre>
    voicedSpeech = isSpeech & isVoiced;
    thisFeature(~voicedSpeech,:) = [];
    thisFeature(:,[featureMap.zerocrossrate,featureMap.shortTimeEnergy]) = [];
    label = repelem(allLabels(ii), size(thisFeature,1));
    features = [features;thisFeature];
    labels = [labels,label];
end
```

Pitch and MFCC are not on the same scale. This will bias the classifier. Normalize the features by subtracting the mean and dividing the standard deviation.

```
M = mean(features,1);
S = std(features,[],1);
features = (features-M)./S;
```

Train Classifier

Now that you have collected features for all 10 speakers, you can train a classifier based on them. In this example, you use a K-nearest neighbor (KNN) classifier. KNN is a classification technique naturally suited for multiclass classification. The hyperparameters for the nearest neighbor classifier include the number of nearest neighbors, the distance metric used to compute distance to the neighbors, and the weight of the distance metric. The hyperparameters are selected to optimize validation accuracy and performance on the test set. In this example, the number of neighbors is set to 5 and the metric for distance chosen is squared-inverse weighted Euclidean distance. For more information about the classifier, refer to fitcknn.

Train the classifier and print the cross-validation accuracy. crossval and kfoldloss are used to compute the cross-validation accuracy for the KNN classifier.

Specify all the classifier options and train the classifier.

```
trainedClassifier = fitcknn(features, labels, ...
   Distance="euclidean", ...
   NumNeighbors=20, ...
   DistanceWeight="squaredinverse", ...
   Standardize=false, ...
   ClassNames=unique(labels));
```

Perform cross-validation.

```
k = 20;
group = labels;
c = cvpartition(group, KFold=k); % 5-fold stratified cross validation
partitionedModel = crossval(trainedClassifier, CVPartition=c);
```

Compute the validation accuracy.

```
validationAccuracy = 1 - kfoldLoss(partitionedModel,LossFun="ClassifError");
fprintf('\nValidation accuracy = %.2f%%\n', validationAccuracy*100);
```

Validation accuracy = 83.65%

Visualize the confusion chart.

```
validationPredictions = kfoldPredict(partitionedModel);
figure(Units="normalized",Position=[0.4 0.4 0.4 0.4])
confusionchart(labels,validationPredictions,title="Validation Accuracy", ...
ColumnSummary="column-normalized",RowSummary="row-normalized");
```

								va	IIda	uon	ACC	sura	Сy						
adrian	6324	179	27	242	72	73	317	61	137	61	14	22	142	190	6	73	234	77.4%	22.6%
alfredo	59	4496	17	96	43	26	256	51	167	24	18	11	152	38	6	30	56	81.1%	18.9%
andre	31	140	6700	64	42	26	125	11	134	65	1	3	136	493	6	50	231	81.1%	18.9%
aris	124	112	32	5563	130	66	246	44	59	136	16	22	164	77	1	101	205	78.4%	21.6%
bruno	71	42	18	180	5401	132	257	25	107	46	10	10	133	71	1	107	226	79.0%	21.0%
dan	87	80	21	146	107	3886	156	30	58	47	9	18	84	73	2	60	192	76.9%	23.1%
fredy	89	116	22	145	113	32	7291	43	53	37	17	5	146	233		97	407	82.4%	17.6%
σ it	36	16	2	50	34	19	92	6821	1	6	114	192	39	54	112	75	111	87.7%	12.3%
S juanpanlo	30	105	15	7	13	5	106	15	4582	24	6	4	27	18	7	1	13	92.0%	8.0%
juanpaulo	30	35	5	63	55	30	122	32	15	6961	9	10	37	58	1	79	144	90.6%	9.4%
	19	21	1	50	35	12	60	198	1	5	3145	208	26	12	98	69	32	78.8%	21.2%
e majo	14	7	3	24	22	9	32	295	1	7	163	4633	13	10	198	59	22	84.1%	15.9%
olguin	25	44	2	73	81	24	143	26	11	21	15	9	6787	21	1	53	72	91.6%	8.4%
oscar	62	36	280	115	63	28	309	39	39	22	7	9	52	16307	1	67	1629	85.5%	14.5%
pao		4		3			1	250		1	98	226		3	3742	25	4	85.9%	14.1%
yestli	44	10	9	101	101	29	128	133		31	45	164	36	121	58	6872	337	83.6%	16.4%
yonathan	53	26	308	103	105	34	277	59	21	35	11	10	50	1756	5	138	14534	82.9%	17.1%
	89.1%	82.2%	89.8%	79.2%	84.2%	87.7%	73.5%	83.9%	85.1%	92.5%	85.0%	83.4%	84.6%	83.5%	88.2%	86.4%	78.8%		
	10.9%	17.8%	10.2%	20.8%	15.8%	12.3%	26.5%	16.1%	14.9%	7.5%	15.0%	16.6%	15.4%	16.5%	11.8%	13.6%	21.2%		
adrian ed andre aris runo dan redy it wan auto mar guin ecar pao yestii than juan paulo mar guin ecar pao yestii than Predicted Class																			

Validation Accuracy

You can also use the Classification Learner app to try out and compare various classifiers with your table of features.

Test Classifier

In this section, you test the trained KNN classifier with speech signals from each of the 10 speakers to see how well it behaves with signals that were not used to train it.

Read files, extract features from the test set, and normalize them.

```
features = [];
labels = [];
numVectorsPerFile = [];

allFeatures = extract(afe,adsTest);
allLabels = adsTest.Labels;

for ii = 1:numel(allFeatures)

    thisFeature = allFeatures{ii};
    isSpeech = thisFeature(:,featureMap.shortTimeEnergy) > energyThreshold;
    isVoiced = thisFeature(:,featureMap.zerocrossrate) < zcrThreshold;

    voicedSpeech = isSpeech & isVoiced;

    thisFeature(~voicedSpeech,:) = [];</pre>
```

```
numVec = size(thisFeature,1);
  thisFeature(:,[featureMap.zerocrossrate,featureMap.shortTimeEnergy]) = [];

label = repelem(allLabels(ii),numVec);

numVectorsPerFile = [numVectorsPerFile,numVec];
  features = [features;thisFeature];
  labels = [labels,label];
end
features = (features-M)./S;
```

Predict the label (speaker) for each frame by calling predict on trainedClassifier.

```
prediction = predict(trainedClassifier,features);
prediction = categorical(string(prediction));
```

Visualize the confusion chart.

```
figure(Units="normalized",Position=[0.4 0.4 0.4 0.4])
confusionchart(labels(:),prediction,title="Test Accuracy (Per Frame)", ...
ColumnSummary="column-normalized",RowSummary="row-normalized");
```

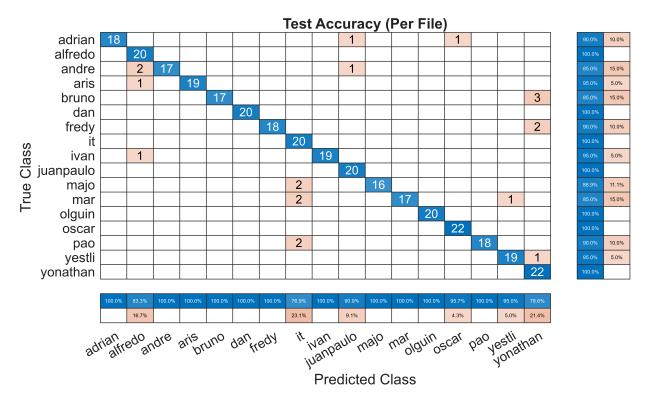
Test Accuracy (Per Frame)																			
adrian	1211	56	11	128	3	26	92	3	64	29	6	7	27	99	10	19	80	64.7%	35.3%
alfredo	41	913	28	25	16	15	127	17	104	9	4	9	40	19		3	21	65.6%	34.4%
andre	32	86	1292	37	15	17	59	3	77	45		1	68	203	2	12	135	62.0%	38.0%
aris	74	95	27	1027	24	39	68	6	47	53	2	4	50	22		20	48	63.9%	36.1%
bruno	74	19	8	72	1250	71	94	2	48	31	2	3	49	103		33	184	61.2%	38.8%
dan	19	16	2	36	13	949	6	6	22	10	3	2	18	1	3	5	1	85.3%	14.7%
fredy	70	52	10	99	52	41	1034	21	91	16	3	4	45	102		55	157	55.8%	44.2%
رم it	11	20	2	39	54	12	54	1507	5	5	64	106	16	41	44	46	59	72.3%	27.7%
S juanpaulo	6	64	9	1	10	15	61	3	788	6	1	1	10	19	7		7	78.2%	21.8%
juanpaulo ∫	12	17		42	22	10	35	11	8	1584	6	3	22	1		24	15	87.4%	12.6%
<u>φ</u> majo	3	9	3	6	6	1	7	106		2	494	128	11	4	57	22	13	56.7%	43.3%
majo mar	16	6	1	40	29	9	35	149	1	10	99	933	13	14	93	81	47	59.2%	40.8%
olgulii	9	102	13	12	5	9	43	2	22	2	5	10	1502	9			15	85.3%	14.7%
oscar	11	11	105	24	2	1	52	6	23	3	1		15	4814	2	7	369	88.4%	11.6%
pao	3	8		6	5	6	8	154	2		42	141	3	5	808	45	9	64.9%	35.1%
yestli	19	10	4	47	35	17	77	48		20	16	82	9	124	36	1541	253	65.9%	
yonathan	10	9	168	11	2	4	55	34	7	4	6	2	5	620	1	8	4047	81.1%	18.9%
1	74.70/	04.00/	70.00/	00.00/	04.00/	70.40/	54.2%	70.50/	00.00/	00.00/	05 50/	05.00/	70.00/	77.00/	70.00/	00.00/	74.40/		
	74.7%	61.2%	76.8%	62.2%	81.0%	76.4%		72.5%	60.2%	86.6%	65.5%	65.0%	78.9%	77.6%	76.0%	80.2%	74.1%		
	25.3%	38.8%	23.2%	37.8%	19.0%	23.6%	45.8%	27.5%	39.8%	13.4%	34.5%	35.0%	21.1%	22.4%	24.0%	19.8%	25.9%		
adrian red ^o ndre aris runo dan redy it wan aulo mar guin escar pao vesti than juan paulo mar guin escar pao vesti than Predicted Class																			

For a given file, predictions are made for every frame. Determine the mode of predictions for each file and then plot the confusion chart.

```
r2 = prediction(1:numel(adsTest.Files));
idx = 1;
```

```
for ii = 1:numel(adsTest.Files)
    r2(ii) = mode(prediction(idx:idx+numVectorsPerFile(ii)-1));
    idx = idx + numVectorsPerFile(ii);
end

figure(Units="normalized",Position=[0.4 0.4 0.4 0.4])
confusionchart(adsTest.Labels,r2,title="Test Accuracy (Per File)", ...
    ColumnSummary="column-normalized",RowSummary="row-normalized");
```



The predicted speakers match the expected speakers for all files under test.

Guardado de archivos .mat

```
% Guardar las variables normalizadas en esa ruta
save("C:\Users\ROG\Downloads\audios\proyecto\mat\archivos1.mat", 'features',
'labels', 'trainedClassifier','M','S');
```

References

[1] Mozilla Common Voice

Copyright 2017-2023 The MathWorks, Inc.