この教材は『手を動かしながらやさしく学べるはじめてのAIデータサイエンスリテラシー』（技術評論社）の実践編として本書の v〜vii ページで紹介している発展的な実践教材です。本書で学んだことを活かして、発展的で面白いプロジェクトに挑戦できます。この教材はPDFで提供しています。必要に応じて印刷し、Wolframノートブックに自分でコードを入力して実行してください。（Wolframノートブックでの配布はしていません。）
なお、ここでは、本書独自のフォーマットを使い、プログラムの入力部分にオレンジの枠，出力部分にブルーの枠をつけてプログラムの部分をわかりやすく表示しています。新規でノートブックを開いた時にはこれらの色枠は付きません。

実践編４：音声データを使った機械学習に挑戦しよう！

第5章で学んだ機械学習の２つの手法を使って、動物の鳴き声の音声データを使った機械学習に挑戦しましょう。１つ目は、教師あり機械学習で動物の鳴き声を分類するプログラムを作ります。さらに、判別器の性能を確認する混同行列についても説明します。２つ目は、教師なし機械学習の手法を使って動物の鳴き声データをグループ分けします。機械は人間と同じように動物ごとにグループに分けられるでしょうか？

＊注意＊この演習は、音声データファイルをWolfram言語にインポートして行います。
外部ファイルをインポートするには、Wolfram Cloudの有料アカウントが必要です。また、この演習のための音声データファイルおよびデータファイルのインポートの方法については本書サポートページを参照して下さい。

（1）教師あり機械学習で鳴き声を聞き当てよう

教師あり機械学習を使って、猫、鳥、牛の3種類の動物の鳴き声を判別するプログラムを作りましょう。
教師あり機械学習の訓練データにするには、各データにそれぞれラベルをつける必要があります。この演習では、ファイルを読み込む際に、それぞれの動物の名前をラベルとしてつけるプログラムで、簡単にラベル付きの訓練データを作っていきます。

1. 準備　訓練データを準備する

本書サポートページからダウンロードしたSampleDataFilesフォルダの中にあるPractice_4_Dataフォルダには、cats、birds、cowsというフォルダに３種類の動物の鳴き声データファイルがそれぞれ10個ずつ入っています。事前にPractice_4_DataフォルダごとWolfram Cloudにアップロードしておきましょう。アップロードの方法については本書サポートページを参照して下さい。

まず、以下を実行して、Practice_4_Dataフォルダにアクセスできるようパスを設定します。この演習では、演習を行うWolframノートブックがあるフォルダの下にPractice_4_Dataフォルダがあることを想定しています。

In[1]=
```
SetDirectory[NotebookDirectory[] <> "Practice_4_Data"]
```
[ディレクトリ…　[ノートブックのディレクトリ

次に、これらのファイルをWolfram言語で扱えるようにするため、**Import**関数を使って読み込みます。 以下は、ファイルを読み込む際に、それぞれの動物の名前をラベルとしてつけるプログラムです。

cats フォルダの中のファイルに「ねこ」、birdsフォルダのファイルには「とり」、cowsフォルダのファイルには「うし」というラベルをつけて**Import**関数でファイルを読み込み、それぞれの動物の訓練データを**neko, tori, ushi** に代入しておきます。**Map**関数を使うと、フォルダ内のファイル１つ１つが**Import[#]**の#に入って、一度に処理できます。

ここでは、最後に ; (セミコロン) をつけて、読み込んだ音声情報を画面に表示しないようにしています。

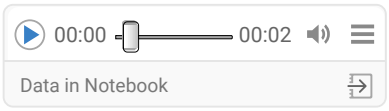
In[2]=
```
neko = Map[Import[#] → "ねこ" &, FileNames["cats/*"]];
       適用 インポート          ファイル名
tori = Map[Import[#] → "とり" &, FileNames["birds/*"]];
       適用 インポート          ファイル名
ushi = Map[Import[#] → "うし" &, FileNames["cows/*"]];
       適用 インポート          ファイル名
```

変数neko, tori, ushi に代入されている音声データがどうなっているかを見てみます。

nekoの1番目の要素を取り出してみましょう。

In[3]=
```
neko[[1]]
```

Out[3]=



→ ねこ

1つの音声データに「ねこ」のラベルが付いていることが確認できます。プレイボタン▶を押すと、このデータの鳴き声を聴くことができます。

練習問題(1)：2番目，3番目の要素はどうなっているでしょうか。また、tori, ushi も上記と同様の方法で確認してみましょう。

各動物10個ずつのデータがあるので、そのうちの７個を訓練データ、残り３個をテストデータにします。ここでは、RandomSample関数を使って、それぞれランダムに訓練データとテストデータに分けます。

In[4]=
```
{trainneko, testneko} = TakeDrop[RandomSample[neko], 7];
                        リスト…    乱数のサンプル
{traintori, testtori} = TakeDrop[RandomSample[tori], 7];
                        リスト…    乱数のサンプル
{trainushi, testushi} = TakeDrop[RandomSample[ushi], 7];
                        リスト…    乱数のサンプル
```

そして、猫、鳥、牛のそれぞれの訓練データ、テストデータをJoin関数でまとめて、traindata, testdata とします。

In[5]=
```
traindata = Join[trainneko, traintori, trainushi];
            繋ぐ
testdata = Join[testneko, testtori, testushi];
           繋ぐ
```


ここで、データの個数を確認しておきましょう。**Map**関数を使って、**Length**関数を**traindata** と**testdata**に対して一度に適用し、要素の数 （データの件数）の結果を出力しています。訓練データが21個、テストデータが9個あることが確認できます。

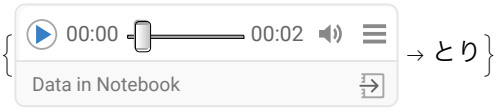
In[6]=
```
Map[Length[#] &, {traindata, testdata}]
```
適用 長さ

Out[6]=
```
{21, 9}
```

RandomSample関数を使って、訓練データからランダムに1つを取り出してみましょう。「音声データ -> ラベル」という形式になっていますね。

In[7]=
```
RandomSample[traindata, 1]
```
乱数のサンプル

Out[7]=



2. 学習　学習モデルを訓練する

1.で準備した訓練データ **traindata**を**Classify**関数に与えて実行すると訓練が始まります。（訓練には少し時間がかかるので、結果が表示されるまで待ちましょう。ここでは、**Classify**関数が学習モデルを自動で選んでいます。）

この実行結果を変数**nakigoe**に代入しておきます。そうすることで、この**nakigoe**は、鳴き声を聞き分けるための分類関数になります。鳴き声を判別するという意味で判別器**nakigoe**と呼びます。

In[8]=
```
nakigoe = Classify[traindata]
```
分類

Out[8]=

Information関数では、判別器**nakigoe**に関する詳しい情報を出力することができます。

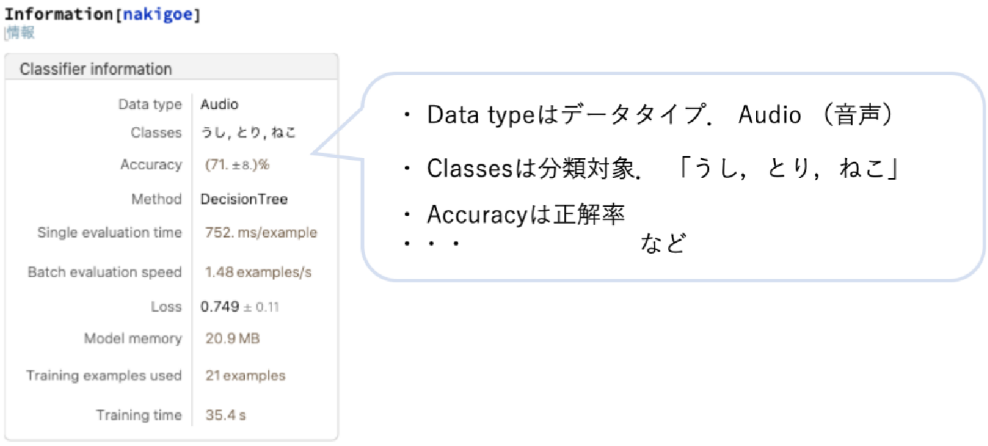


図1 ClassifierFunction関数のInformation関数での出力例

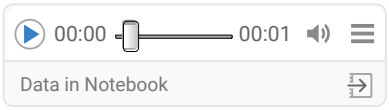
考えてみよう：この判別器nakigoeの正解率は何％でしたか？また、このnakigoeは訓練データをいくつ使っているでしょうか？（ヒント：Information関数の出力結果Classifier informationの “Training examples used”に書かれています）

3. テスト　未知のデータを使って判定する

判別器**nakigoe**でテストデータの鳴き声を判定してみましょう。**testdata**のデータは9個あるので、ここでは2番目のデータを判定してみます。**testdata[[2]]**で、2番目のデータの中身を確認します。

In[9]=
```
testdata[[2]]
```

Out[9]=



→ ねこ

testdata[[2,1]]とすると、上で実行した**testdata**の2番目の要素のうち、鳴き声データだけを取り出すことができます。**testdata[[2,1]]**を判別器**nakigoe**に与え、判別してみましょう。

In[10]=
```
nakigoe[testdata[[2, 1]]]
```

Out[10]=

とり

「とり」という結果になりました。2番目のデータは「ねこ」なので、判別器**nakigoe**では正しく判定できなかったようです。

練習問題(2)：テストデータは全部で9個あります。上記の方法で他のテストデータも判定してみましょう。結果はどうなるでしょうか？

判別器の性能と混同行列

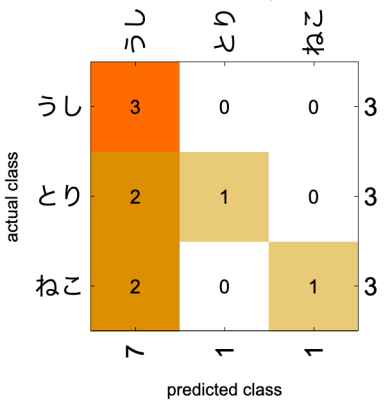
判別器の性能は、**ClassifierMeasurements**関数を使って調べることができます。テストデータでの結果は以下のようになりました。

In[11]=
```
ClassifierMeasurements[nakigoe, testdata, "Report"]
```
|分類測度

Out[11]=

Classifier Measurements

Classifier method	DecisionTree
Number of test examples	9
Accuracy	(56. ±18.)%
Accuracy baseline	(33. ±17.)%
Geometric mean of probabilities	0.246 ± 0.13
Mean cross entropy	1.40 ± 0.52
Single evaluation time	773. ms/example
Batch evaluation speed	1.3 examples/s



上記の出力結果で、最後に表示されている格子図（図2)は、縦軸がラベルの答え(actual class)、横軸が判別器が予測した答え(predicted class)です。判別器がどれくらい正解したか、何をどう間違ったのかを見ることができます。機械学習ではこの格子図を「**混同行列**」と言います。

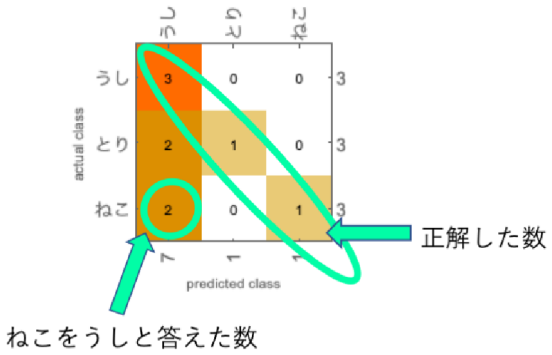


図2 判別器の性能をみる混同行列

例えば、図2の対角線上の数字3,1,1は、ラベルの答えと判別器が予測した答えが一致した、いわゆる正解した数です。図2では、縦軸の「うし」と横軸の「うし」が交わる格子に3という数字が入っているので、牛の鳴き声は3つとも正解していることがわかります。しかし、2行目の「とり」（正解）に対しては、1列目の「うし」に数字2が入っているので、「とり」の鳴き声のうち2つを、「うし」に間違えているようです。同様に、3行目の「ねこ」の行を見ると、1列目の「うし」に2が入っているので、2つは「うし」と間違えていますね。

考えてみよう：判別器nakigoeの性能について、Information関数やClassifierMeasurements関数の結果から考察してみましょう。例えば、格子図からどんなことが言えるでしょうか？どのような間違いが起こりやすいでしょうか？また、より良い判別器を作るにはどうしたら良いかを考えてみましょう。

（2）教師なし学習で動物の鳴き声をグループ分けしてみよう

（1）の演習と同じ音声データを使って、今度は動物の鳴き声をその特徴でグループ分けをしてみましょう。第5章5-5節で学んだ教師なし機械学習です。

1. 準備　データを準備する

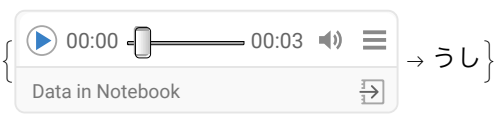
（1）の1.で作成した**neko, tori, ushi**をすべて**Join**関数でまとめて、変数**animalvoice**に代入します。

In[12]=
```
animalvoice = Join[neko, tori, ushi];
```
繋ぐ

RandomSample関数を使って**animalvoice**から適当な1つを取り出してみると、以下のように「音声データ -> ラベル」という形式になっていることが確認できます。

In[13]=
```
RandomSample[animalvoice, 1]
```
乱数のサンプル

Out[13]=

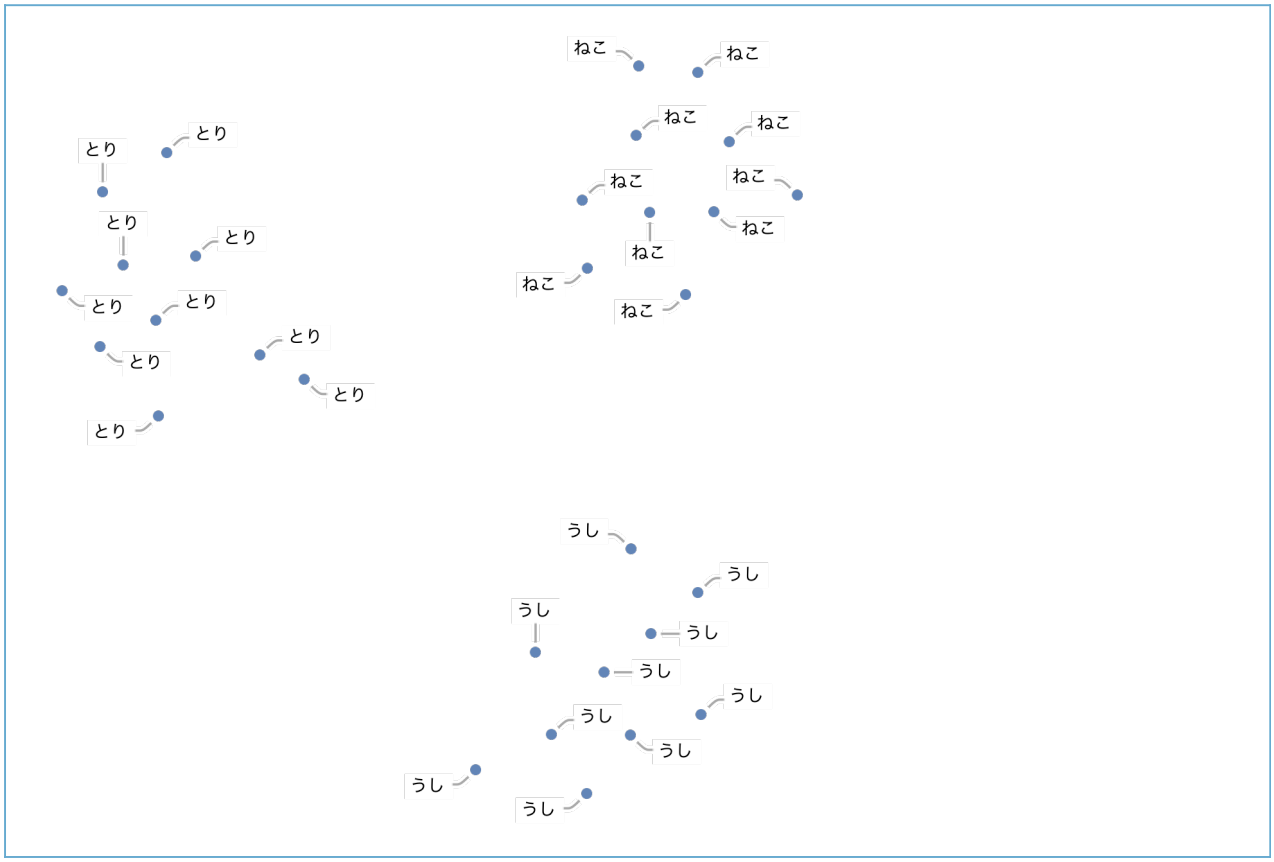


2. 学習　データに学習モデルを適用する

1. で準備した音声データを学習モデルに与えます。ここでは、第5章の演習5-6でも使用した**FeatureSpacePlot**関数で、音声データから抽出された特徴を数値化した値を二次元空間にプロットします。特徴が似ているものは近くの位置に描かれます。同じ動物同士が近い位置に描かれているでしょうか？

In[14]=
```
FeatureSpacePlot[animalvoice, LabelingFunction → Callout]
```
特徴空間プロット　ラベル付け関数　コールアウト

Out[14]=



ワンポイント：「教師なし機械学習」は答えを与えることができません。上記の「とり」「ねこ」「うし」の名前は、音声データを「教師なし機械学習」でグループ分けした後に、描画した後のラベルとして使われています。（名前の情報はモデルの学習には使用していないことに注意しましょう）

3. 活用　学習結果から新しい特徴を見つける

2. で出力されたグラフは私たちの考える動物の種類に分かれていて、このデータではグループ分けがうまくできたように見えます。でも、さらにグラフをよく見てみましょう。縦軸や横軸に注目してデータを辿ると何か新しい発見があるかもしれません。

考えてみよう：教師なし学習では、どのように鳴き声をグループ化しているのでしょうか。第4章で学んだ音声データの正体や特徴を踏まえて、考えてみましょう。