scikit-learn

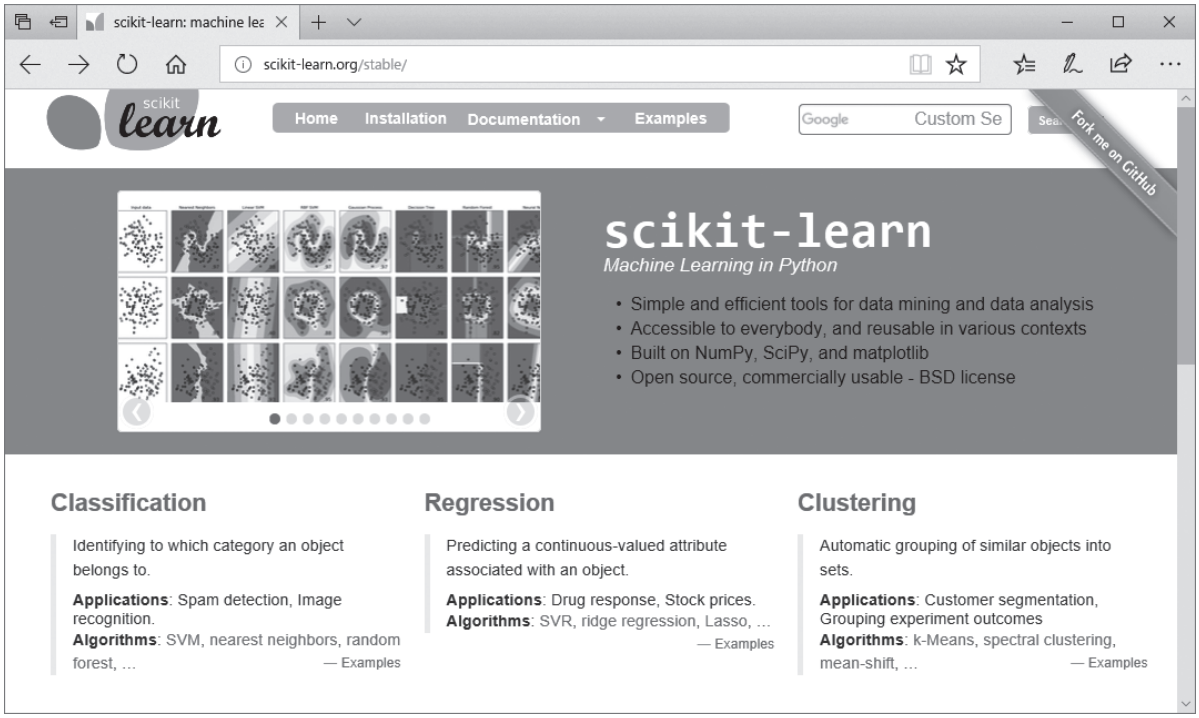
2日目 Part 1 scikit-learn

今日は、scikit-learnについて調べましょう。scikit-learn（サイキットラーン）は、オープンソース（BSDライセンス）のライブラリで、機械学習に必要な回帰、分類、クラスタリングなどのアルゴリズムを備えています。

また、Pythonの数値計算ライブラリであるNumPyやSciPyを内部で利用しているようです。公式サイトには、scikit-learnについての詳しいドキュメントもあるので、scikit-learnを使う前に一度覗いてみた方がよいでしょう（**図1**）。

図1●scikit-learn公式サイト

http://scikit-learn.org



ここでもう一度、機械学習の種類をおさらいしておきます。機械学習の手法は、大きく分けると次の種類があります。

▶教師あり学習

教師あり学習の特徴は、学習データに正解ラベルがついていることです。この正解付き

データを元に、データの特徴、パターンを学習します。そして、正解ラベルのないデータに対して、そのデータがどの正解ラベルと一致するかを予測します。

教師あり学習は、さらに正解ラベルの種類によって「回帰」(regression)と「分類」(classification)に分けられます。

▶教師なし学習

教師なし学習は、正解ラベルのないデータに対して、特徴や共通点、パターンなどを発見する手法です。正解ラベルがないので、学習データをそろえる手間は少なくなりますが、どのような特徴やパターンが導き出されるのかは、やってみなければわからないという問題があります。

教師なし学習の手法には「クラスタリング」(clustering)や「次元圧縮」(dimensionality reduction)などがあります。

▶その他(強化学習など)

機械学習には「教師あり学習」「教師なし学習」の、どちらにも分けることができない手法も登場しています。例えば「強化学習」と呼ばれる手法では、行動した結果に応じて報酬が与えられるようにすることで、徐々により良い行動へと進むように学習します。

▶scikit-learnの機械学習アルゴリズム

scikit-learnには、機械学習アルゴリズムが数多く実装されていますが、大まかに次の4つのグループに分けることができます。

回帰 (regression)

実数値をデータで学習し、実数値を予測する。scikit-learnが搭載するアルゴリズムには、SGD回帰、LASSO回帰などがある。

分類 (classification)

正解ラベルとそのデータを学習し、データに対してのラベルを予測する。scikit-learnが搭載するアルゴリズムには、カーネル近似、k近傍法などがある。

クラスタリング (clustering)

データの似ている部分をグループにして、データの特徴やパターンを発見する。scikit-learnが搭載するアルゴリズムには、k平均法、スペクトラルクラスタリングなどがある。

次元圧縮（dimensionality reduction）

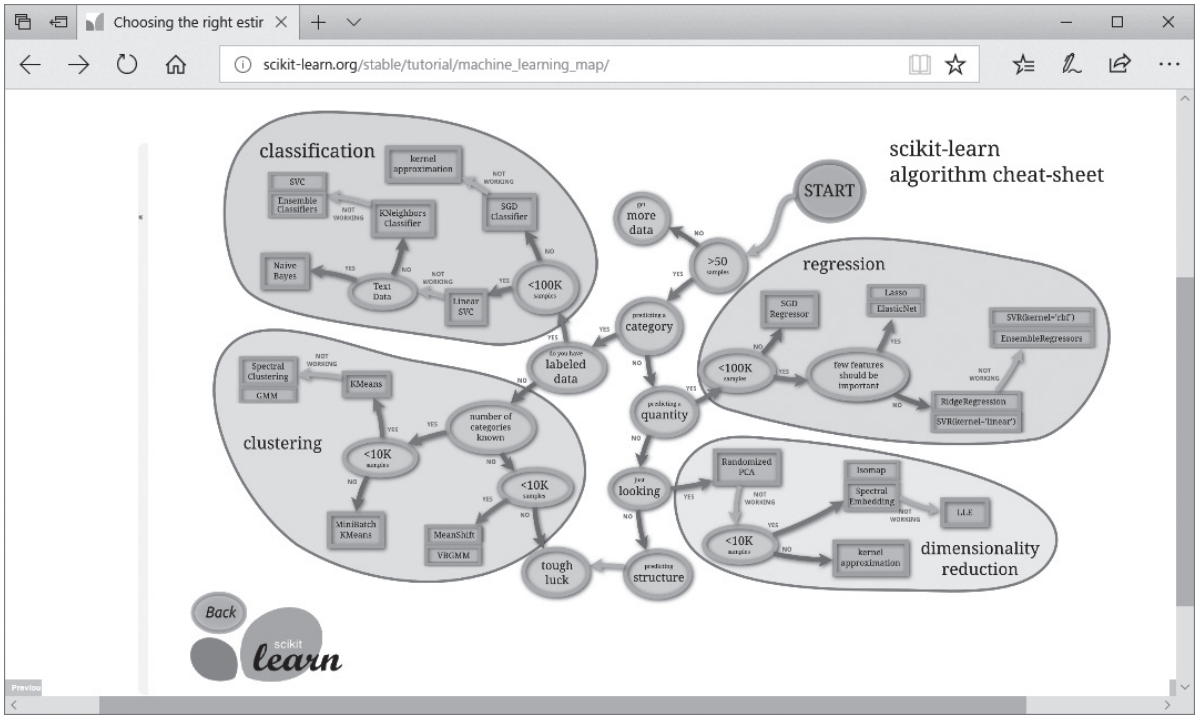
データの次元を削減して、固有の構造を見つけ出す手法。または、他の手法の入力に利用する。scikit-learnが搭載するアルゴリズムには、主成分分析（PCA）、カーネルPCAなどがある。

▶どのアルゴリズムが最適か

このように、scikit-learnには多くのアルゴリズムが実装されています。実際には、上記に掲載したアルゴリズムの倍以上あります。これだけ多いと、さすがに心が折れそうになります…が、そんな私を勇気づけてくれる図が、scikit-learnのサイトにありました（**図2**）。

図2●アルゴリズムマップ

http://scikit-learn.org/stable/tutorial/machine_learning_map/



このマップは、質問にYes、Noと回答していくと、最適な機械学習アルゴリズムへたどり着くチャートです。例えば、「START」→「50個以上のサンプル（学習データ）があります

か?」という質問に対して、「NO」なら、「もっとデータを用意してください」と怒られます。「カテゴリ予測ですか?」という質問に「YES」と答えると、「学習データにラベルが付いていますか?」という質問に進み、「Yes」なら「教師あり学習アルゴリズムの分類グループ」へ、「NO」なら「教師なしアルゴリズムのクラスタリンググループ」へ進みます。

もっとも、最適なアルゴリズム(を実装したクラス)がわかっても、まだどうやって利用すれば良いのかわかりません。ただ、この中で多少でもイメージできるアルゴリズムと言えば、回帰 (regression)アルゴリズムでしょうか。そこで本格的な機械学習アルゴリズムへ進む前に、scikit-learnで最も単純だと思われる回帰アルゴリズムを試してみましょう。

2日目 Part 2 回帰分析

回帰分析は、「結果データ」と「結果に影響を及ぼすデータ」の関係性を統計的に求める手法です。結果データを「目的変数」、結果に影響を及ぼすデータを「説明変数」と呼びます。

scikit-learnには、この回帰分析を行うモデルとして「線形回帰モデル」が搭載されています。線形回帰とは、すべてのデータにできるだけフィットする線を引くことで、予測を行うモデルです。線の式は「最小2乗法」を用いて求めます。

また、説明変数が1つの場合を「単回帰分析」、複数の場合を「重回帰分析」と呼びます。scikit-learnでは、どちらの回帰分析も行うことができます。ここでは、単回帰分析を行ってみましょう。

▶単回帰分析

scikit-learnでは、「linear_model.LinearRegression」というクラスを用いることで、線形回帰モデルを作ることができます。「linear_model.LinearRegression」を使ってモデルを生成するときの書式は、**図1**のようになります。

実行はJupyter Notebookで行いますので、起動しておきましょう。最初に、必要なライブラリをインポートします。おなじみのNumPy、matplotlib、Pandasの他に、「sklearn」の「linear_model」をインポートします。

図1●linear_model.LinearRegressionの書式

```
sklearn.linear_model.LinearRegression(fit_intercept=True, norma
lize=False,copy_X=True, n_jobs=1)
```

引数	説明
fit_intercept	切片を求める計算を含めるか否か
normalize	説明変数を事前に正規化するか否か
copy_X	メモリー内で上書きしないようにXを複製してから実行するか否か
n_jobs	計算に使うジョブの数。 ー1にすると、利用可能なCPUをすべて使う

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from pandas import DataFrame
        from sklearn import linear_model
        %matplotlib inline
```

pandasのDataFrame形式で、X軸とY軸のデータをわかりやすい感じで用意します。

```
In [2]: X = DataFrame([0,1,2,3,4,5])
        Y = DataFrame([3,2,5,7,6,10])
```

とりあえず、どのような散布図になるかmatplotlibでプロットしてみましょう（**図2**）。

```
In [3]: plt.scatter(X, Y, color='red')
```

線形回帰は、すべてのデータにできるだけフィットする線を引くことで予測を行うモデルです。まずは、目測で回帰直線を引いてみます。大体、こんな感じになるはずです（**図3**）。

XとYのデータを公式に当てはめると回帰係数や切片が求まるので、そこから計算して回帰直線を引くことができますが、その計算をscikit-learnのLinearRegressionで行います。

では、scikit-learnのLinearRegressionで、線形回帰モデルを生成します。

図2●データの散布図

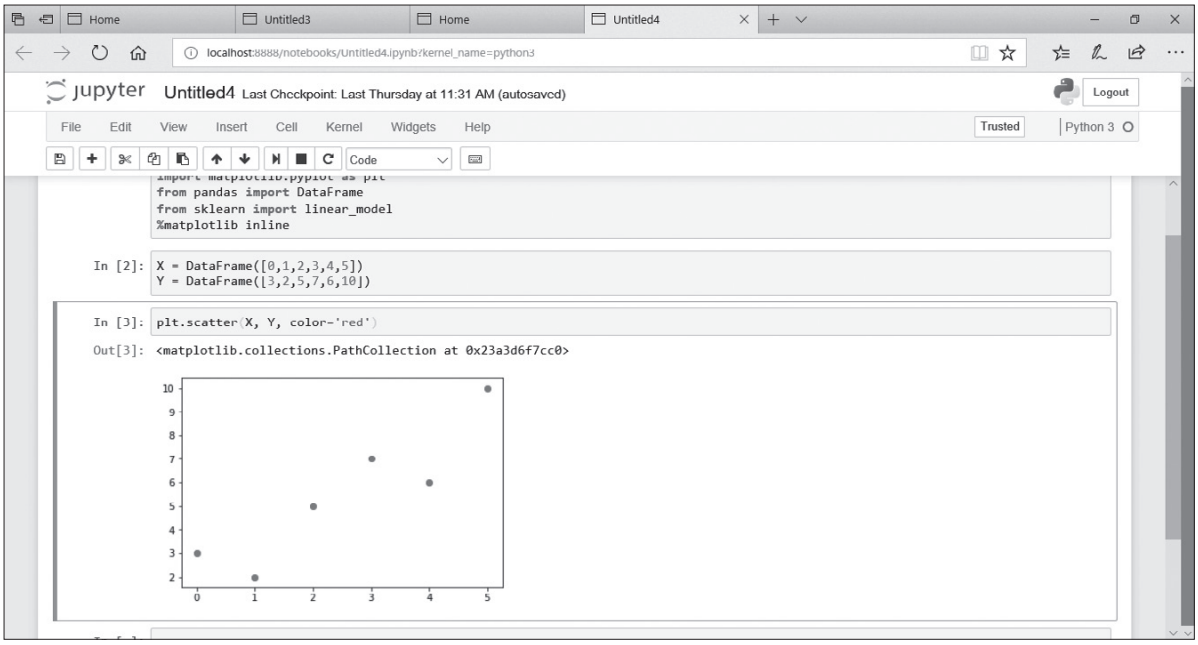
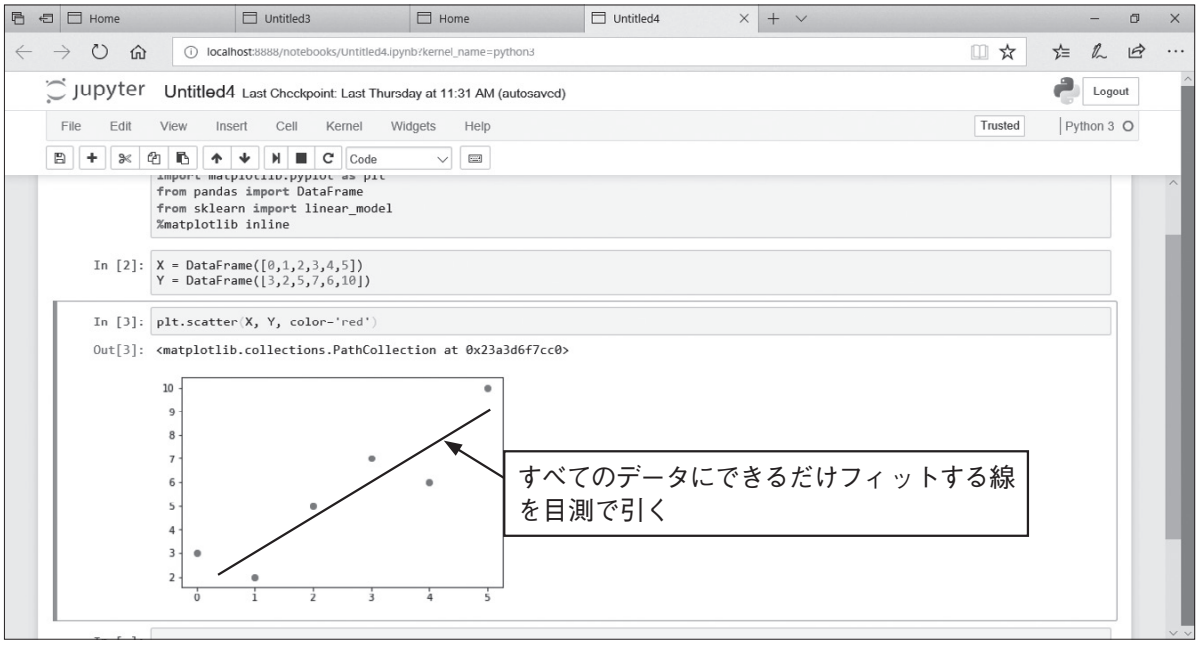


図3●目測で引いた回帰直線




```
In [4]: model = linear_model.LinearRegression()
```

作成したモデルに、XとYのデータを渡します。学習（機械学習では「訓練」と呼ぶ）には、fit関数を使います。実行すると、どのようなパラメータで線形回帰モデルが学習したか確認の表示が出力されます。

```
In [5]: model.fit(X, Y)
Out[5]: LinearRegression(copy_X=True, fit_intercept=True, n_jo
bs=1, normalize=False)
```

これで、XとYのデータを学習した線形回帰モデルが完成しました。

NumPyのarange関数を使い、Xの最小値から最大値まで0.01刻みの配列（array([0. , 0.01, 0.02,…）を生成します。

```
In [6]: tmp = np.arange(X.min(), X.max(),0.01)
```

NumPyのnewaxis関数を使い、X座標を2次元配列（array([[0.],[0.01],[0.02],…）に変換します。

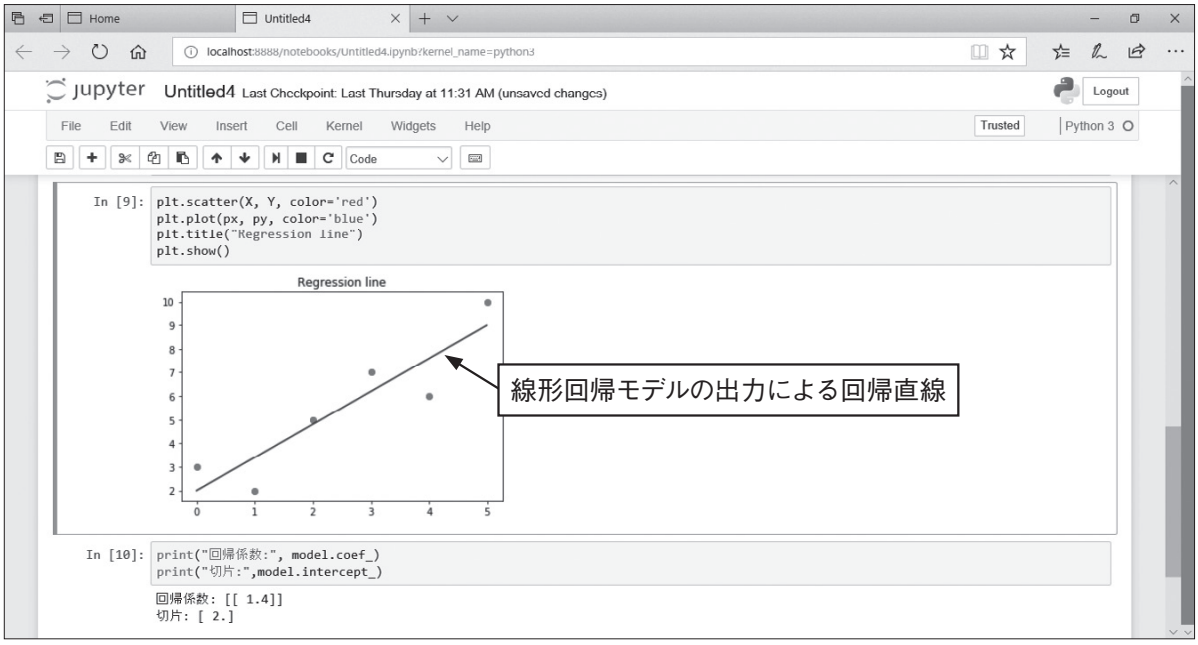
```
In [7]: px = tmp[:,np.newaxis]
```

X座標を線形回帰モデルのpredict関数に渡して、Y座標を作ります。

```
In [8]: py = model.predict(px)
```

では、元のデータと作成した回帰直線を一緒に表示してみます（**図4**）。

図4●回帰直線の表示




```
In [9]: plt.scatter(X, Y, color='red')
        plt.plot(px, py, color='blue')
```

```
        plt.title("Regression line")
        plt.show()
```

うまくいきました。もし、導き出した回帰係数や切片が知りたいときは、係数はcoef_属性に、切片はintercept_属性に格納されています。

```
In [10]: print("回帰係数:", model.coef_)
         print("切片:",model.intercept_)
         回帰係数: [[ 1.4]]
         切片: [ 2.]
```

scikit-learnのLinearRegressionクラスを使うと、アルゴリズムの仕組みさえ知っていれば、実際の計算は勝手にやってくれることがわかります。拍子抜けしてしまうほど簡単です。これはなかなか「スゴイ」かも…。

2日目 Part 3 機械学習用のデータ

scikit-learnの使い方は、なんとなくイメージできました。ただ、本格的な機械学習の実験を行うためには、それなりのデータが必要になります。

ネット上には、機械学習用のサンプルデータがいくつも落ちていますが、まずはscikit-learnに付属しているデータを使うことにします。

まず、iris（アイリス）という「アヤメ」（花）のデータセットと、digits（デジッツ）という手書き数字のデータセットを調べました。

▶irisデータセット

irisデータセットは、統計分析や機械学習のサンプルとして有名で、機械学習の入門にはなくてはならないデータセットです。irisデータセットの読み込みは、専用のload_iris関数があるので、これを利用します。

```
In [1]: from sklearn.datasets import load_iris
```

```
        iris_dataset = load_iris()
```

scikit-learnのドキュメントによると、ここで取得したirisデータは、scikit-learnで定義されているBunchクラスのオブジェクトになっています（**図1**）。

図1●load_irisの説明

http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html

データはタプルで'data'、'target'などのキーで取得するようです。'DESCR'キーでirisデータセットの説明を表示してみます（**図2**）。

```
In [2]: print(iris_dataset['DESCR'])
```

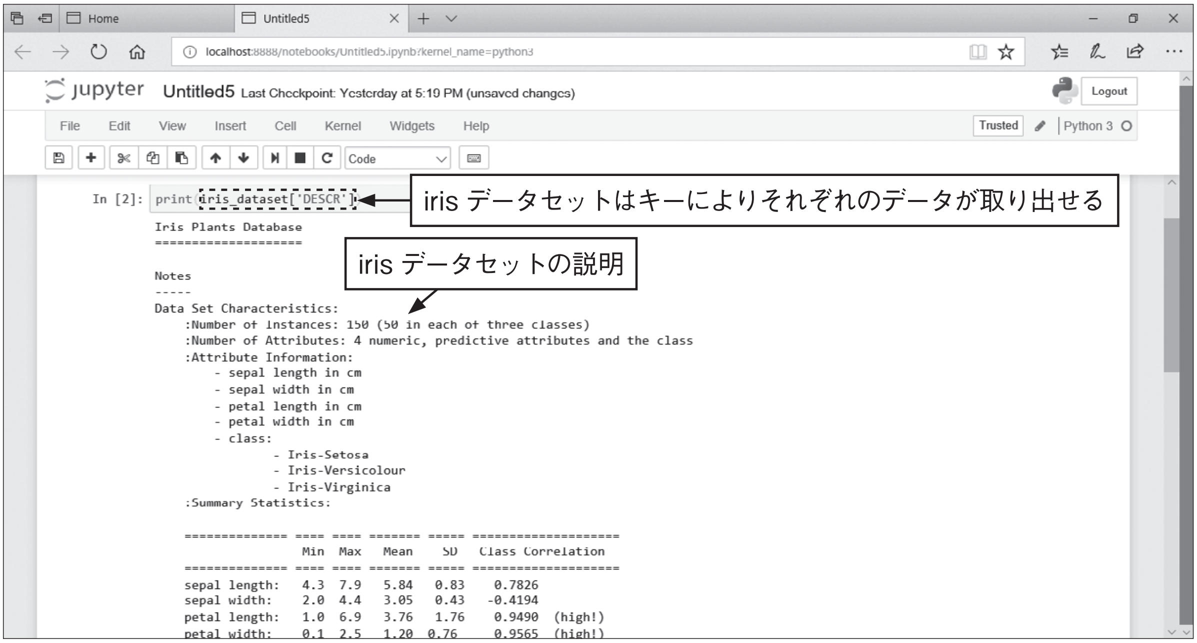
表示を見ると、アヤメのデータは「150個」、単位は「センチメートル」と書いてあります。「Iris-Setosa」（アイリス-セトナ）「Iris-Versicolour」（アイリス-バーシクル）「Iris-Virginica」（アイリス-バージニカ）の3種類のアヤメのSepal（がく片）とPetal（花弁）のlength（長さ）およびwidth（幅）を計測したデータになっています。

では、'DESCR'以外のキーに、何があるか調べましょう。

```
In [3]: print(iris_dataset.keys())
        dict_keys(['data', 'target', 'target_names', 'DESCR',
'feature_names'])
```

'DESCR'以外に'data'、'target'、'target_names'、'feature_names'というキーがあり

図2●irisデータセットの説明



ます。それぞれのデータを表示してみます。

'data'キーは、アヤメの特徴データです。順番は、がく片の長さ、がく片の幅、花弁の長さ、花弁の幅です。

```
In [4]: print(iris_dataset['data'])
        [[ 5.1  3.5  1.4  0.2]
         [ 4.9  3.   1.4  0.2]
         [ 4.7  3.2  1.3  0.2]
...
```

'target'キーを表示すると、150個の数字が出てきました。0、1、2の3種類の数字が50個ずつあるので、アヤメの種類を示しているようです。

```
In [5]:print(iris_dataset['target'])
       [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
        1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
2 2 2 2 2 2 2 2 2 2
        2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
2 2 2 2 2 2 2 2 2 2
        2 2]
```

それぞれの名前を'target_names'キーで確認します。0がsetosaのデータ、1がversicolorのデータ、2がvirginicaのデータということですね。

```
In [6]:print(iris_dataset['target_names'])
      ['setosa' 'versicolor' 'virginica']
```

最後の'feature_names'キーは何でしょうか。featureは「特徴」のことなので、各特徴データに付けられた名前のようです。

```
In [7]:print(iris_dataset['feature_names'])
       ['sepal length (cm)', 'sepal width (cm)', 'petal length
(cm)', 'petal width (cm)']      ['setosa' 'versicolor' 'virgin
ica']
```

PandasのDataFrameで表示してみましょう。

```
In [8] :import pandas as pld
        pld.DataFrame(iris_dataset.data, columns=iris_dataset.
feature_names)
```

irisデータセットが、どのような構成かわかりました（**図3**）。

▶手書き数字のデータセット

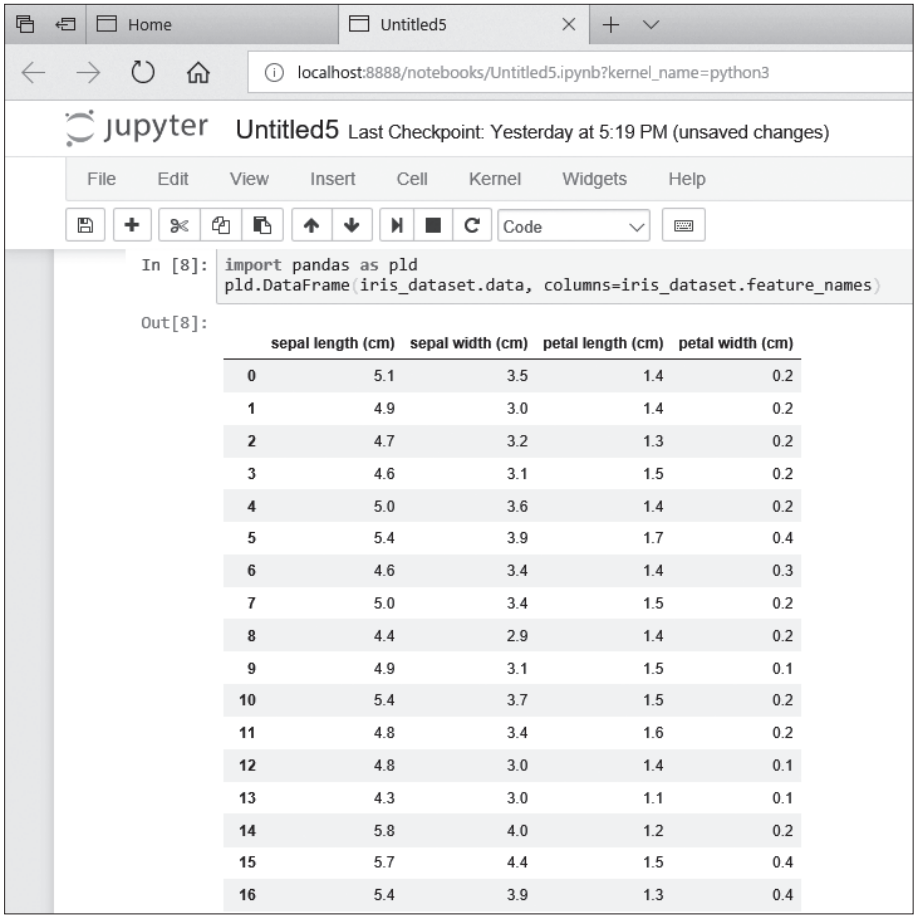
図4の学習用データは、「digits」という手書き数字の画像データと、各画像に付けられたラベルデータです。

オリジナルのデータは、「MNIST」という名前で「http://yann.lecun.com/exdb/mnist/」から入手できますが、scikit-learnに付属しているdigitsデータセットは、MNISTの簡易版です。digitsのデータセットについては、scikit-learnのサイトにある「The Digit Dataset」に説明があります（**図5**）。

では、digitsデータセットをロードしてみましょう。

```
In [9] :from sklearn.datasets import load_digits
        digits = load_digits()
```

図3●irisデータセット



	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1
10	5.4	3.7	1.5	0.2
11	4.8	3.4	1.6	0.2
12	4.8	3.0	1.4	0.1
13	4.3	3.0	1.1	0.1
14	5.8	4.0	1.2	0.2
15	5.7	4.4	1.5	0.4
16	5.4	3.9	1.3	0.4

図4●digitsのデータセット

http://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html

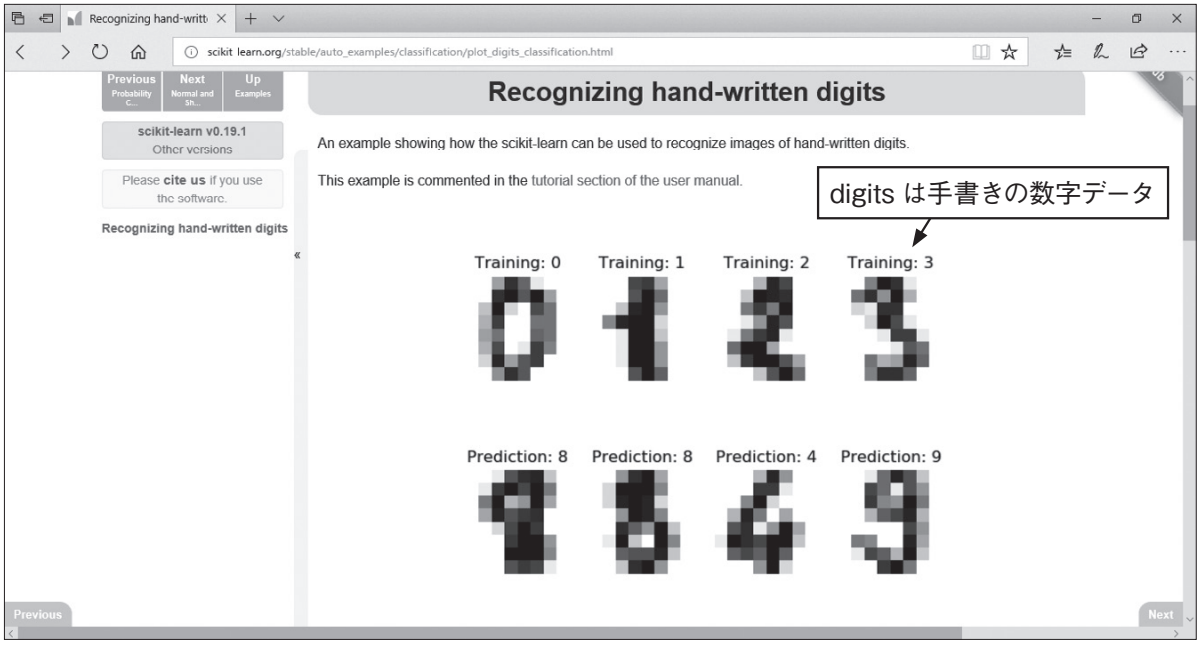
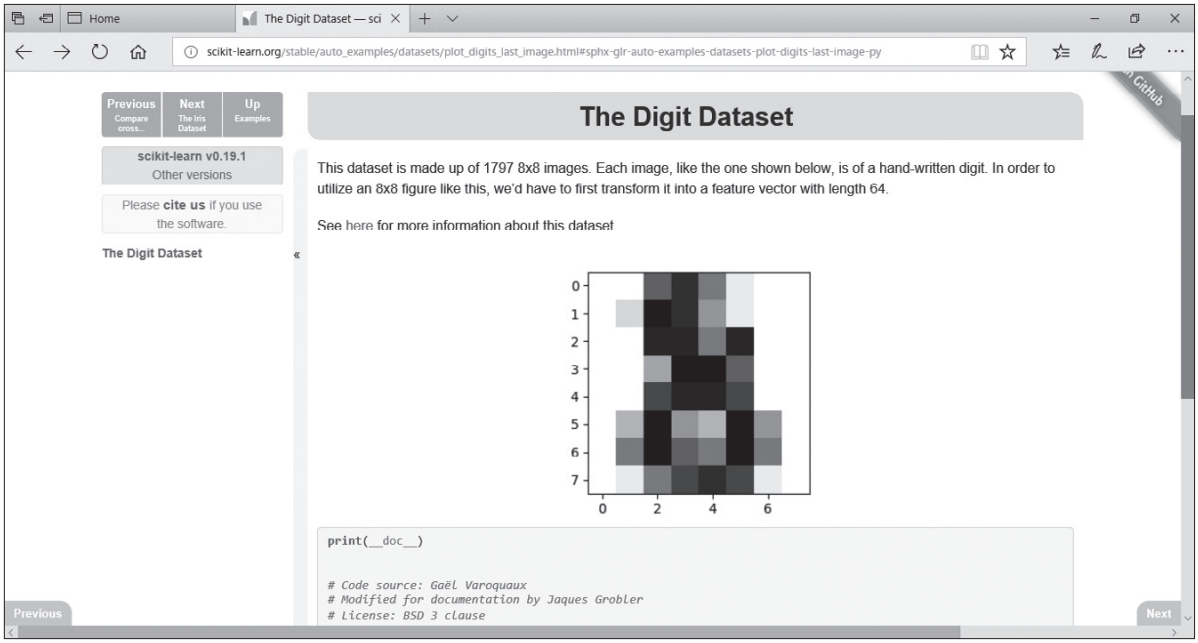


図5●digitsのデータセットの説明

http://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html#sphx-glr-auto-examples-datasets-plot-digits-last-image-py



データと、その数を表示してみます。

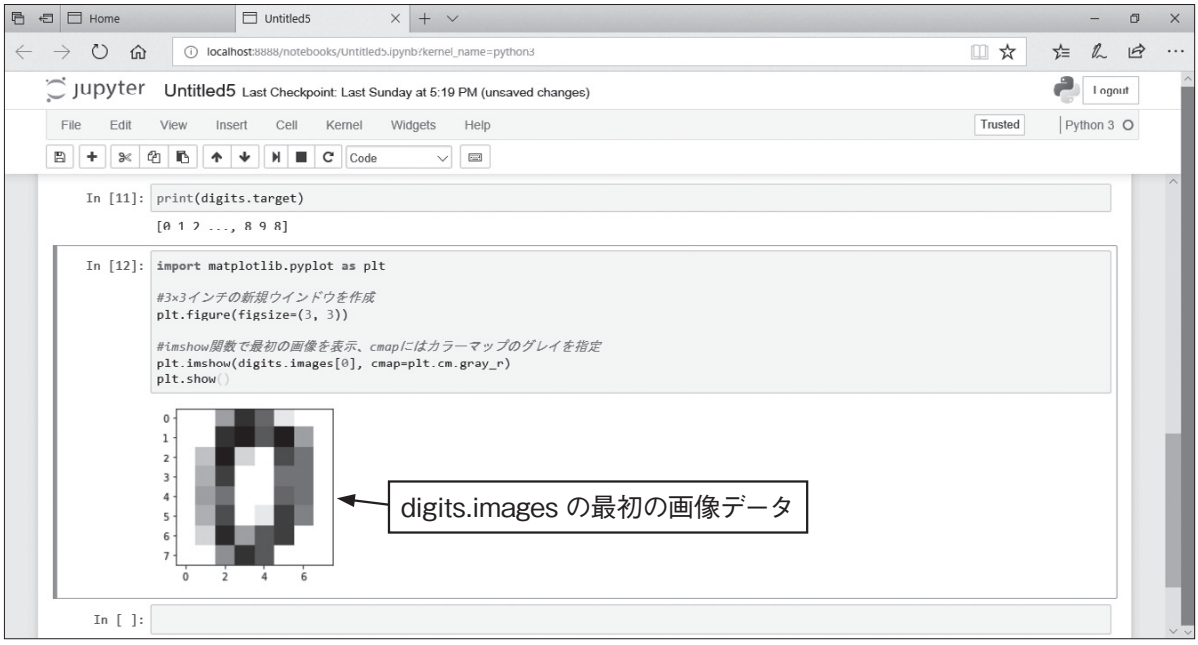
```
In [10]: print(digits.data)
         [[  0.   0.   5. ...,   0.   0.   0.]
         [  0.   0.   0. ...,  10.   0.   0.]
         [  0.   0.   0. ...,  16.   9.   0.]
          ...,
         [  0.   0.   1. ...,   6.   0.   0.]
         [  0.   0.   2. ...,  12.   0.   0.]
         [  0.   0.  10. ...,  12.   1.   0.]]

In [11]: print(digits.data.shape)
         (1797, 64)
```

データ1件あたり8×8=64個の値がNumPyの配列になっていて、データ件数は1797件あることがわかります。1797件分の正解ラベル(0 ～ 9)は、targetに入っています。

```
In [12]: print(digits.target)
```

図6●最初の画像データを表示




```
[0 1 2 ..., 8 9 8]
```

大体、digitsがどのようなデータセットかわかりました。

「The Digit Dataset」のページを見ていると、digits.imagesに8×8のピクセルデータがあるようなので、matplotlibの勉強も兼ねて、imshow関数で画像を描画してみましょう（**図6**）。

```
In [13]: import matplotlib.pyplot as plt

         # 3×3インチの新規ウインドウを作成
         plt.figure(figsize=(3, 3))

         # imshow関数で最初の画像を表示
         # cmapにはカラーマップのグレイを指定
         plt.imshow(digits.images[0], cmap=plt.cm.gray_r)
         plt.show()
```

今日は、ここまでにします。明日は、このデータセットを使って機械学習のアルゴリズムを試しましょう。