データベース演習 （Advanced Database Exercise）

１０．分類器

URL: http://www.kkaneko.jp/cc/dbenshu/index.html

**概要 Abstract**

今日の授業では、機械学習，自動分類に関する演習を行う．

Today's class is exercise on data classification

■ SVM kernel

■ Nearest Neighbor

■ Random Forests

**演習 (Exercises)**

今日の授業で使用するデータセット（データセット名: digits）の出典と著作権表示

出典：http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits   
===  
E. Alpaydin, Fevzi. Alimoglu   
Department of Computer Engineering   
Bogazici University, 80815 Istanbul Turkey   
alpaydin **'@'** boun.edu.tr

* **ステップ1 (Step 1)**

①　スタートメニューの「**Anaconda3 (64bit)**」の下の「**spyder**」を起動する．右下に IPython コンソール・ウインドウがある．

　　Launch “spyder” in the “Anaconda3 (64bit)” in a start menu. There is an IPython console windows in right lower.



* **ステップ3 (Step 3)**

**データフレーム d のデータを使用したい。**

**SVM を用いて機械学習する． SVM の詳細はこの授業では触れない。** Do classification using SVM

1. **機械学習する．まずはLinear SVM を用いる．Linear SVM のオブジェクトを作り classifier という」名前ののオブジェクトに格納する．Linear SVM の詳細はこの授業では触れない。**

　　Do classification using SVM. A classifier is stored into variable named 'classifier'

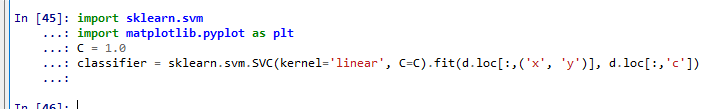
import numpy as np

import sklearn.svm

import matplotlib.pyplot as plt

C = 1.0

classifier = sklearn.svm.SVC(kernel='linear', C=C).fit(d.loc[:,(**'x', 'y'**)], d.loc[:,**'c'**])

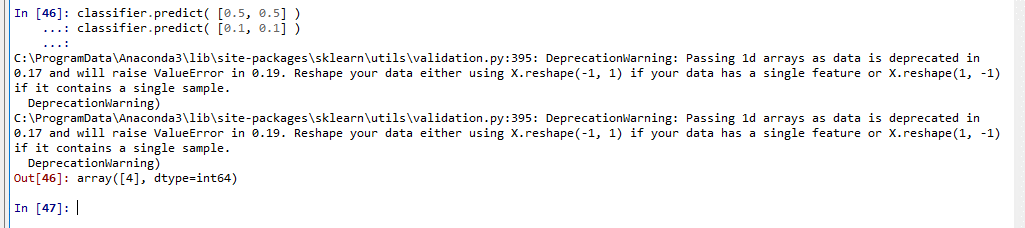


**Liear SVMのオブジェクト（分類機）を使ってみる．動作確認のため．**Use the classifier

動作確認してみる．エラーメッセージが出なければOK.

classifier.predict( [0.5, 0.5] )

classifier.predict( [0.1, 0.1] )



1. **準備．機械学習の結果を表示したい範囲のメッシュグリッドを作る**

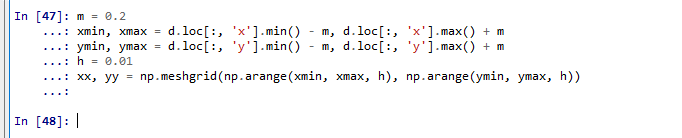
m = 0.2

xmin, xmax = d.loc[:, 'x'].min() - m, d.loc[:, 'x'].max() + m

ymin, ymax = d.loc[:, 'y'].min() - m, d.loc[:, 'y'].max() + m

h = 0.01

xx, yy = np.**meshgrid**(np.arange(xmin, xmax, h), np.arange(ymin, ymax, h))

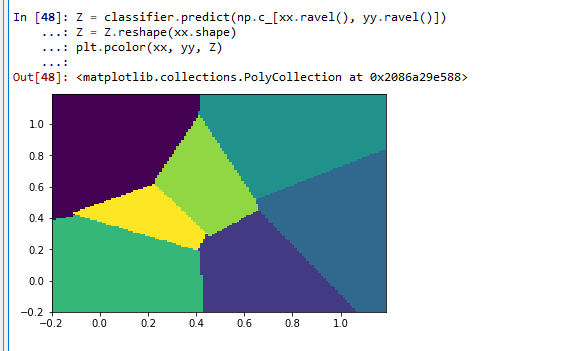


**今作成したメッシュグリッドを使って、プロット** (plot the classifier)

Z = classifier.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

**plt.pcolor(xx, yy, Z)**

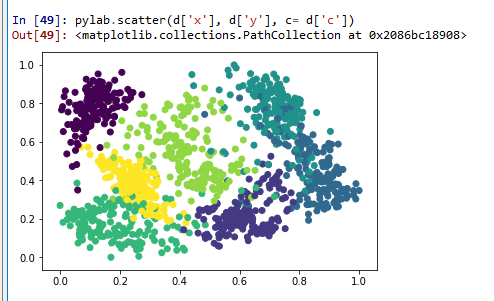


元のデータと見比べてみる．

　元データ（下図）：　すきま**あり**

　機械学習後のデータ（上図）：　すきま**なし**

pylab.scatter(d['x'], d['y'], c= d['c'])



* **ステップ4 (Step 4)**

**次の３つを比べてみる**

・classifier: linear kernel SVM (線形カーネル)

・classifier2: RBF kernel SVM (RBFカーネル)

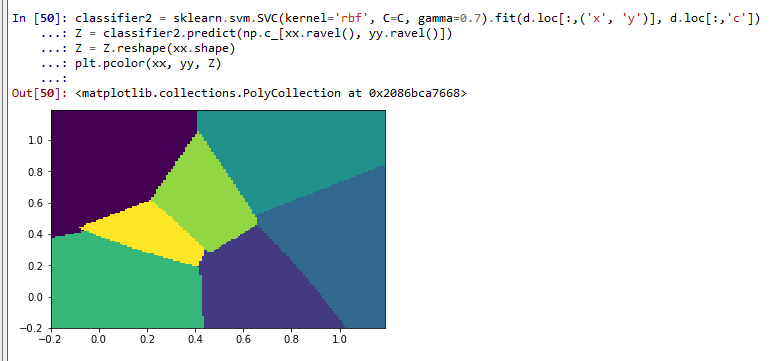
・classifier3: polynomial (degree 3) kernel SVM (次数3の多項式カーネル)

**classifier2** = sklearn.svm.SVC(kernel='**rbf**', C=C, **gamma=0.7**).fit(d.loc[:,(**'x', 'y'**)], d.loc[:,**'c'**])

Z = **classifier2**.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

**plt.pcolor(xx, yy, Z)**

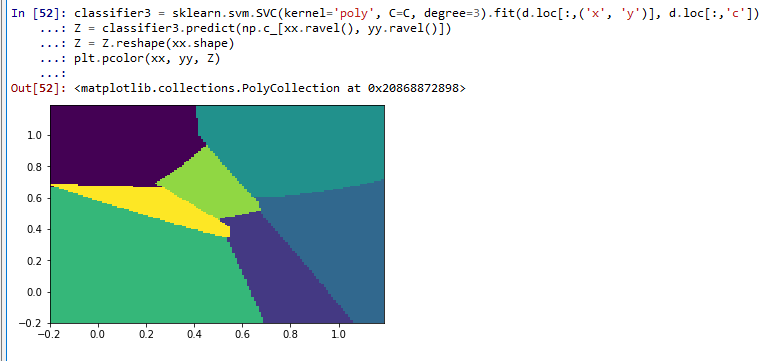


**classifier3** = sklearn.svm.SVC(kernel='**poly**', C=C, **degree=3**).fit(d.loc[:,(**'x', 'y'**)], d.loc[:,**'c'**])

Z = **classifier3**.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

**plt.pcolor(xx, yy, Z)**



**ステップ5 (Step 5)**

分類器の表示

分類器の表示

分類器の表示

**今度は最近傍を用いる． Nearest Neighbot Classification**

**1. 準備**

import sklearn.neighbors



**2. 最も近い1点による分類 Classifiation using the nearest point**

**classifier4** = sklearn.neighbors.KNeighborsClassifier(n\_neighbors=1, weights='uniform').fit(d.loc[:,(**'x', 'y'**)], d.loc[:,**'c'**])

Z = **classifier4**.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

**plt.pcolor(xx, yy, Z)**



分類器の表示

**3. 最も近い3点による分類 Classifiation using the nearest point**

**classifier5** = sklearn.neighbors.KNeighborsClassifier(n\_neighbors=**3**, weights='uniform').fit(d.loc[:,(**'x', 'y'**)], d.loc[:,**'c'**])

Z = **classifier5**.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

**plt.pcolor(xx, yy, Z)**



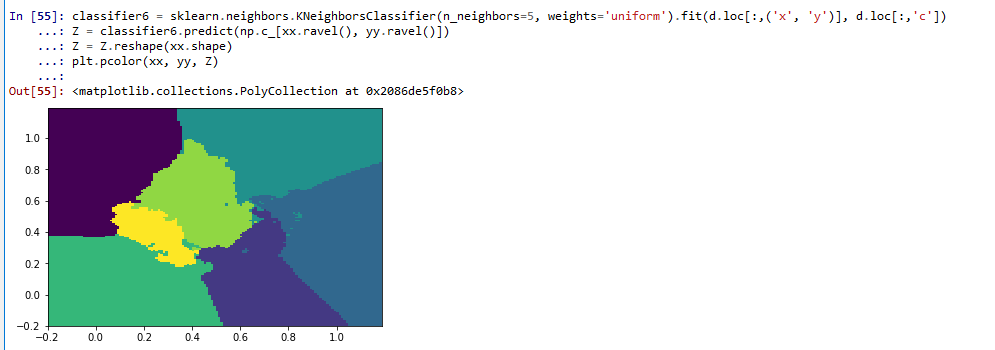
**4. 最も近い5点による分類 Classifiation using the nearest point**

**classifier6** = sklearn.neighbors.KNeighborsClassifier(n\_neighbors=**5**, weights='uniform').fit(d.loc[:,(**'x', 'y'**)], d.loc[:,**'c'**])

Z = **classifier6**.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

**plt.pcolor(xx, yy, Z)**



* **ステップ6 (Step 6)**

**今度は，Random Forestsを用いる．** Random Forest Classification

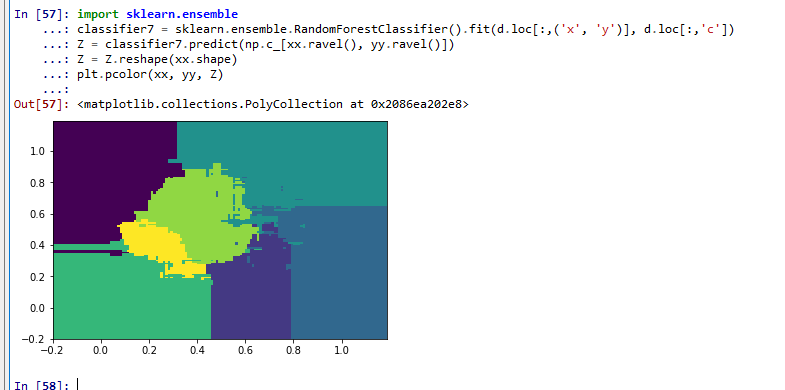
**import sklearn.ensemble**

**classifier7** = sklearn.ensemble.RandomForestClassifier().fit(d.loc[:,(**'x', 'y'**)], d.loc[:,**'c'**])

Z = **classifier7**.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

**plt.pcolor(xx, yy, Z)**



classifier7.feature\_importances\_



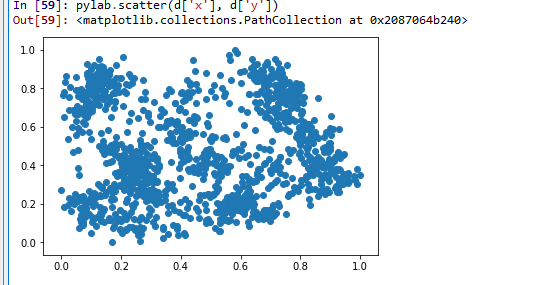
**ここから別テーマになります**

**■ ステップ７ (Step ７)**

1. **次のようなデータを使います．データフレーム d の2列 x, y だけを使います．**

**使用するデータのプロット** Plot the source data

pylab.scatter(d['x'], d['y'])

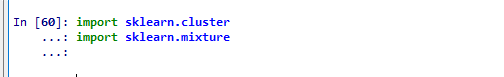


**この中から，固まり（「クラスタ」という）を見つける．このことを「教師なし学習」と言ったりする．**

1. **準備**

import sklearn.cluster

import sklearn.mixture

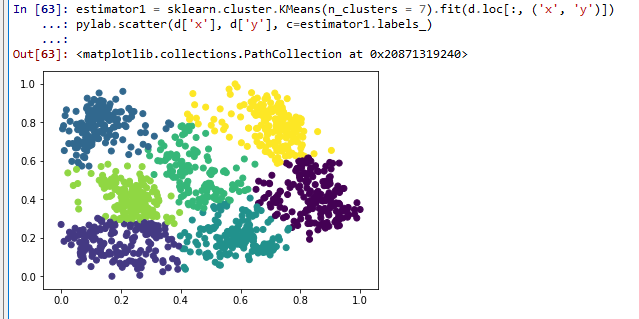


**3. 教師なしで分類する。k-means アルゴリズムを用いる** Classification without learning. Use k-means algorithm

n\_clusters = 7 は「作りたいクラスタのクラスタ数が7である」ことを指定している。このとき、分類ラベルは 0, 1, 2, 3, 4, 5, 6 の7通りになる。

estimator1 = sklearn.cluster.KMeans(n\_clusters = 7).fit(d.loc[:, (**'x', 'y'**)])

pylab.scatter(d['x'], d['y'], c=estimator1.labels\_)

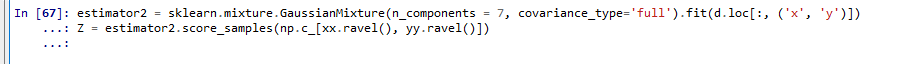


1. **今度は Gaussian Mixture Model（混合ガウスモデル）を用いる**

Classification without learning. Use GMM.

**estimator2** = sklearn.mixture.GaussianMixture(n\_components = 7, covariance\_type='full').fit(d.loc[:, (**'x', 'y'**)])

Z = **estimator2**.score\_samples(np.c\_[xx.ravel(), yy.ravel()])



**確認表示する**

plt.pcolor(xx, yy, Z.reshape(xx.shape))

