演習：k-means (sklean)

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import cluster, preprocessing, datasets
from sklearn.cluster import KMeans
```

In [2]:

```
# scikit-learnに組み込まれている、ワインのデータセットを読み込む
# https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html
wine = datasets.load_wine()
```

In [3]:

```
X = wine.data
y = wine.target
```

データの中身を確認

In [4]:

```
print(X.shape)
print(y.shape)
print(type(X))
```

```
(178, 13)
(178,)
<class 'numpy.ndarray'>
```

In [5]:

```
print(wine.feature_names)  # 説明変数
```

```
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'fla
vanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue', 'od28
0/od315_of_diluted_wines', 'proline']
```

In [6]:

```
print(wine.target_names)  # 目的変数 (分類するクラス数)
```

```
['class_0' 'class_1' 'class_2']
```

学習

In [7]:

```
model = KMeans(n_clusters=3)  # k-meansのモデルを作成する (クラスタ数を3に設定)
```

In [8]:

```
labels = model.fit_predict(X)  # クラスタ分析を行う
```

In [9]:

```
print(labels)
```

```
[0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0 1 1
 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 2 1 2 2 1 2 2 1 1 1 2 2 0
 1 2 2 2 1 2 2 1 1 2 2 2 2 2 1 1 2 2 2 2 2 1 1 2 1 2 1 2 2 2 1 2 2 2 2 1 2
 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 1 1 1 1 2 2 2 1 1 2 2 1 1 2 1
 1 2 2 2 2 1 1 1 2 1 1 1 2 1 2 1 1 2 1 1 1 1 2 2 1 1 1 1 1 2]
```

モデルの評価

In [10]:

```
df = pd.DataFrame({'labels': labels})
```

In [11]:

```
def species_label(theta):
    """分類されたクラスタの数値に応じてクラス名を付与する"""
    if theta == 0:
        return wine.target_names[0]
    if theta == 1:
        return wine.target_names[1]
    if theta == 2:
        return wine.target_names[2]
```

In [12]:

```
df['species'] = [species_label(theta) for theta in wine.target]  # speciesが正解となるラベル
```

In [13]:

```
pd.crosstab(df['labels'], df['species'])  # 予測と正解の集計
```

Out[13]:

species labels	class_0	class_1	class_2
0	46	1	0
1	13	20	29
2	0	50	19

考察：
labels0が、class_0に対応しているように見える。これに関しては、うまく分類できている。
labels1は、まったく分類ができていない。
labels2は、class_1に対応しているように見えるが、実際はclass_2だったりしているのもいくらかある。
モデルを改善させるには、labels1の分類をもっとうまくできるよう説明変数を減らす等の工夫が必要になると思われる。