ハロートレーニング
急がば学べ

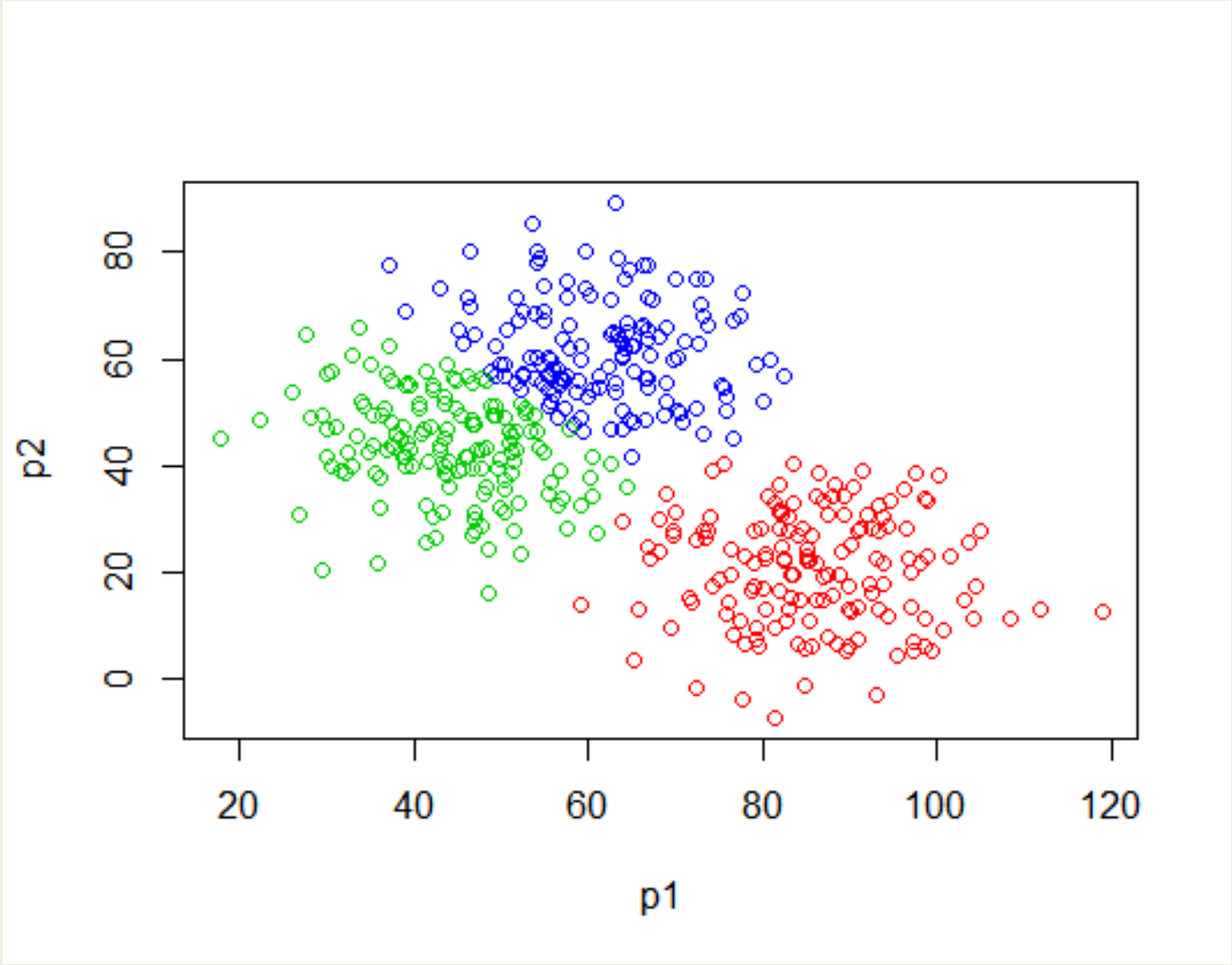
技能向上訓練
データサイエンス
プログラミングコース

Python演習
クラスタリング

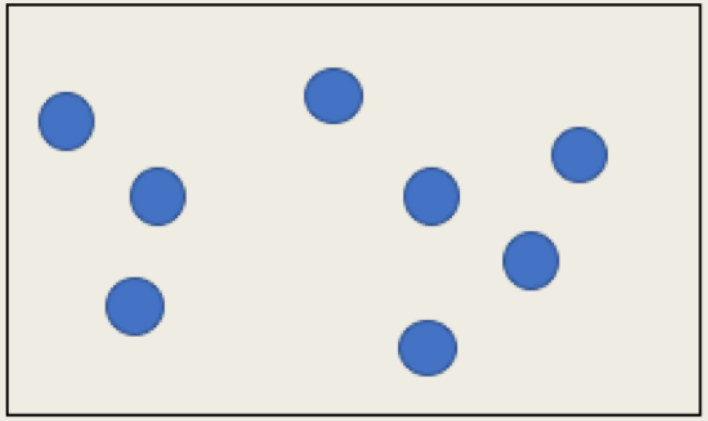
教師なし学習
『クラスタリング』

入力データをグループ分けをする。

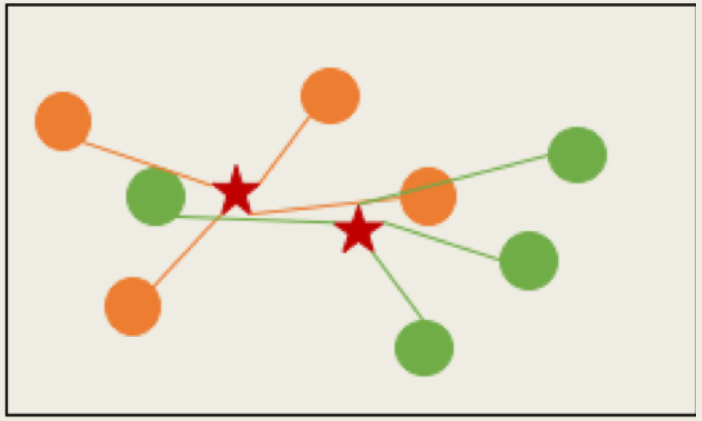
グループの群を、AIが作り出すところが、分類との違いである。



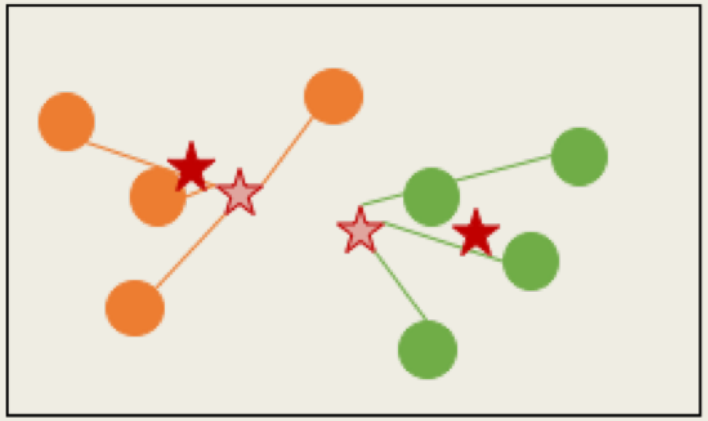
k-means法



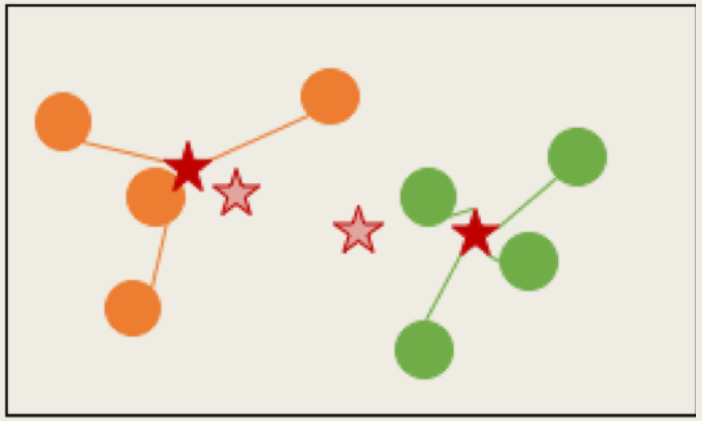
(a)



(b)



(c)



(b)

1.各点xiに対してランダムにクラスタを割り振る

2.各クラスタに割り当てられた点について重心を計算する

3.各点について上記で計算された重心からの距離を計算し、距離が一番近いクラスタに割り当て直す。

4. 2.と3.の工程を、割り当てられるクラスタが変化しなくなるまで行う

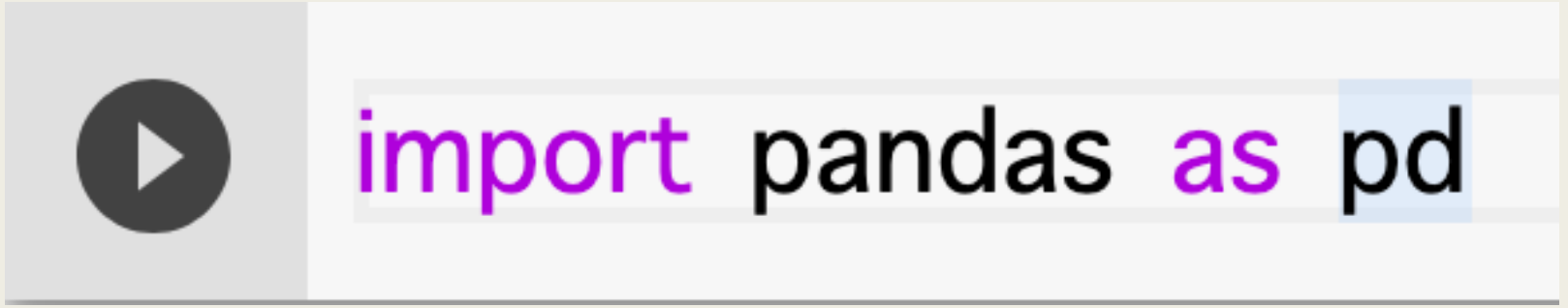
データのダウンロード

- https://github.com/GIbson5527/AI_ensyu
- 上のサイトから、ensyu1をダウンロードし、自身のGoogleDriveに保存する。
- MyDriveに保存したら、フォルダ内の、ensyu1.ipynbを開く

アヤメのクラスタリング

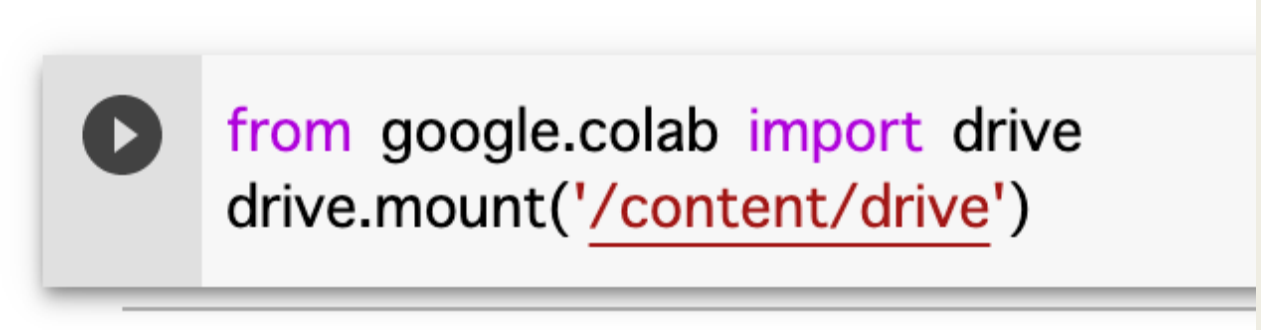
- アヤメのデータセットを使用
- ３種類のアヤメの分類を行う

ライブラリのインポート

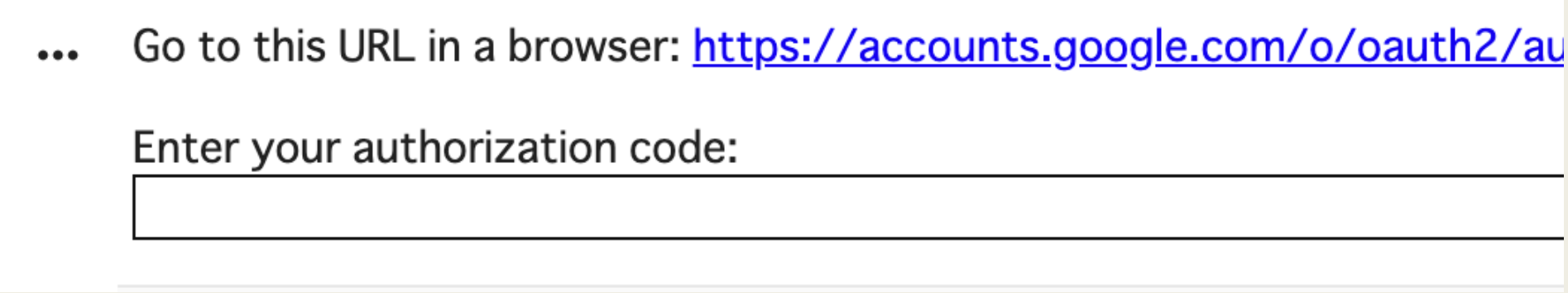


Google Driveをマウント

- Google Colabratoryから、自分のGoogleDriveにアクセスできるようにする。



- 実行後に表示されるURLをクリック。「許可」を選択すると表示されるコードを下の欄に入力する。



アヤメのデータのロード

- GoogleDriveに保存した、アヤメのデータを読み込む

```
# アヤメのデータを読み込み、dfに格納
df = pd.read_csv(filepath_or_buffer="/content/drive/My Drive/csv/iris.csv")
#データの確認
display(df.head())
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

データの整形

- クラスタ算出に不必要な、「Species」列を取り除いたものを「train_X」に格納する。

```
# アヤメのデータをtrain_Xにコピー
train_X = df
# train_Xから、Species列を取り除く
train_X = train_X.drop('Species',axis = 1)
# データの確認
train_X.head()
```

	SepalLength	SepalWidth	PetalLength	PetalWidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2

モデルの作成

```
# KMean法のライブラリをインポート
from sklearn.cluster import KMeans
# クラスタリングのモデル作成
## n_clusters = クラスタ数
kmodel = KMeans(n_clusters=3,random_state = 0)
kmodel
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=0, tol=0.0001, verbose=0)
```

クラスタリングを実行

- クラスタリングを実行し、各データのクラスタ番号を得る

```
# クラスタリングを実行
# 各データに、番号を割り振る
kmodel.fit(train_X)
# 割り振った番号をtrain_Yに格納
train_Y = kmodel.labels_
# 結果を確認
print(train_Y)
```

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

結果の確認

- 算出したクラスタ番号と元のデータを結合する。

```
# dfをdf_resultにコピー
df_results = df
# df_resultsに、列「クラスタ番号」を追加し、クラスタリングの結果を格納
df_results['クラスタ番号'] = train_Y

# 結果の確認
#pd.set_option('display.max_rows', 150)
display(df_results)
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species	クラスタ番号
0	5.1	3.5	1.4	0.2	Iris-setosa	1
1	4.9	3.0	1.4	0.2	Iris-setosa	1
2	4.7	3.2	1.3	0.2	Iris-setosa	1

散布図で確認する

```
# 散布図で確認
plt.scatter(train_X[train_Y == 0]['SepalLength'],
            train_X[train_Y == 0]['PetalLength'],
            c='green',
            label='cluster_0')
plt.scatter(train_X[train_Y == 1]['SepalLength'],
            train_X[train_Y == 1]['PetalLength'],
            c='yellow',
            label='cluster_1')
plt.scatter(train_X[train_Y == 2]['SepalLength'],
            train_X[train_Y == 2]['PetalLength'],
            c='red',
            label='cluster_2')
plt.grid()
plt.legend(loc="upper left")
plt.show()
```

演習: 都道府県のクラスタリング

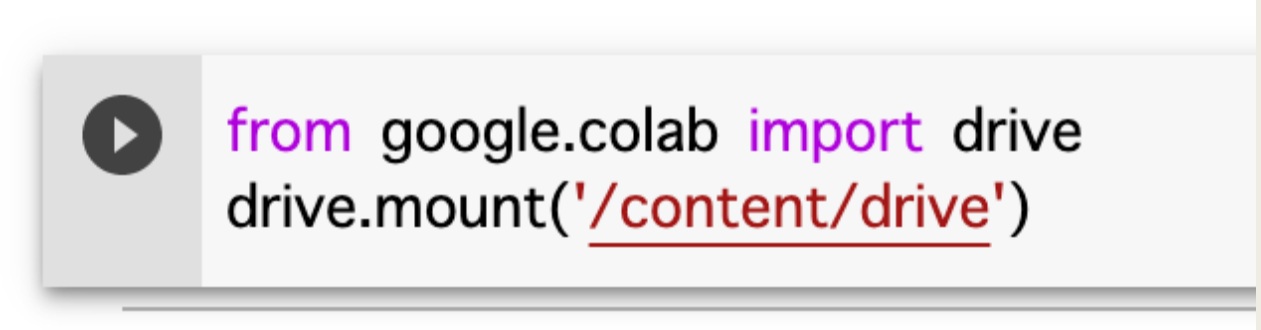
- それでは実際に、オープンデータを用いてクラスタリングを行います。今回は各都道府県の物価データを用いて、類似している県をいくつかのグループに分類してみましょう。

ライブラリの読み込み
解析に使用するライブラリをインポート

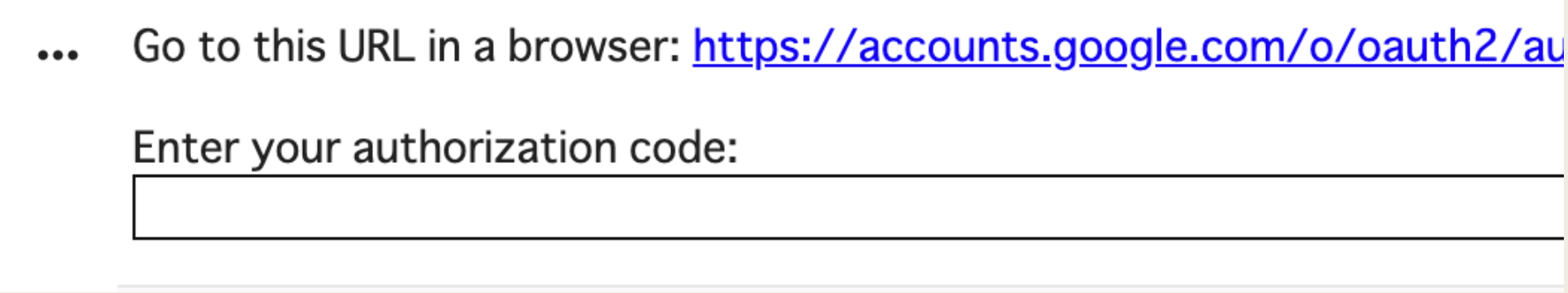
```
[ ]  # Pandasライブラリを、pdという別名で使用できるように宣言する
     # matplotlib.pyplotライブラリを、pltという別名で使用できるように宣言する
     import pandas as pd
     import matplotlib.pyplot as plt
```

Google Driveをマウント

- Google Colabratoryから、自分のGoogleDriveにアクセスできるようにする。



- 実行後に表示されるURLをクリック。「許可」を選択すると表示されるコードを下の欄に入力する。



データの準備

- GoogleDriveに保存した、都道府県データを読み込む

```
df_price_index= pd.read_csv('/content/drive/My  Drive/ensyu1/price_index.csv')
df_price_index.head(15)
```

- データの基礎統計量を確認

```
df_price_index.describe()
```

データの可視化

- 10項目のうち、「食料」と「水道光熱費」のデータを用いて、解析を進めます。まずは散布図による可視化を行って、データの分布を確認します。

```
# 都道府県、食料、水道光熱費の列だけを抽出
df_train = df_price_index[['都道府県', '食料', '水道光熱費']]
df_train.head(15)
```

```
# プロットの準備(x軸:食料、y軸:水道光熱費)
plt.scatter(df_train['食料'], df_train['水道光熱費'])
# X軸、Y軸のラベルを設定
plt.xlabel('Shokuryo')
plt.ylabel('Suido_Konetsu')
```

クラスタリング

- これからすべてのデータをいくつかのクラスタに分割します。まずは、先ほどのデータから都道府県の列を除外して、食料、水道光熱費だけのデータを作成します。

```
# DataFrameをDeep Copy
train_X = df_train.copy(deep=True)
# データから'都道府県'を除外
train_X = train_X.drop('都道府県',axis=1)
train_X.head(15)
```

モデルを生成

- 次にk平均法(k-means)で、47都道府県をクラスタリングするためのモデルを生成します。クラスタ数はとりあえず6に指定します。

```
# k平均法(k-means)のライブラリをインポート
from sklearn.cluster import KMeans
# k-meansのモデルを生成
## n_clusters = クラスタ数
## random_state = 乱数のシードを固定(デフォルト値: None)
kmodel = KMeans(n_clusters=6, random_state=0)
kmodel
```

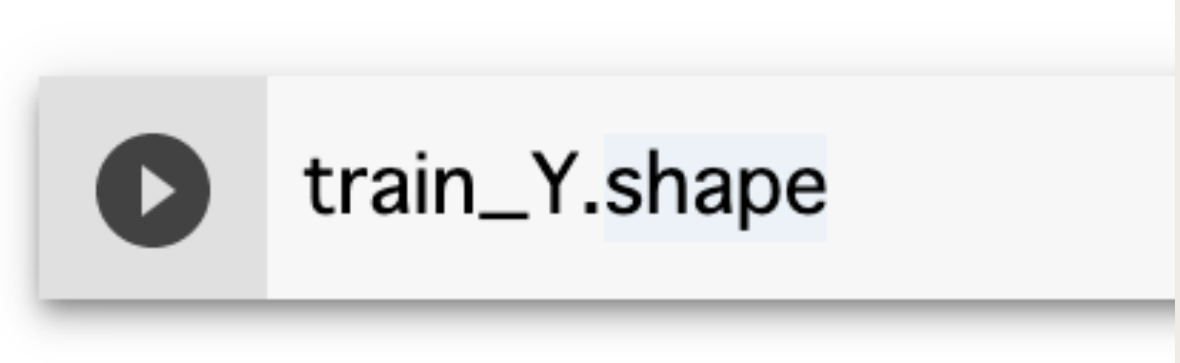
クラスタリングを実行

```
# クラスタリングを実行
kmodel.fit(train_X)
train_Y = kmodel.labels_
print(train_Y)
```

```
[2 3 2 5 3 2 3 5 1 4 4 0 4 1 1 0 0 4 4 5 4 1 1 1 5 0 1 1 5 0 0 2 3 0 3 0 3
 3 0 3 3 2 0 3 5 5 0]
```

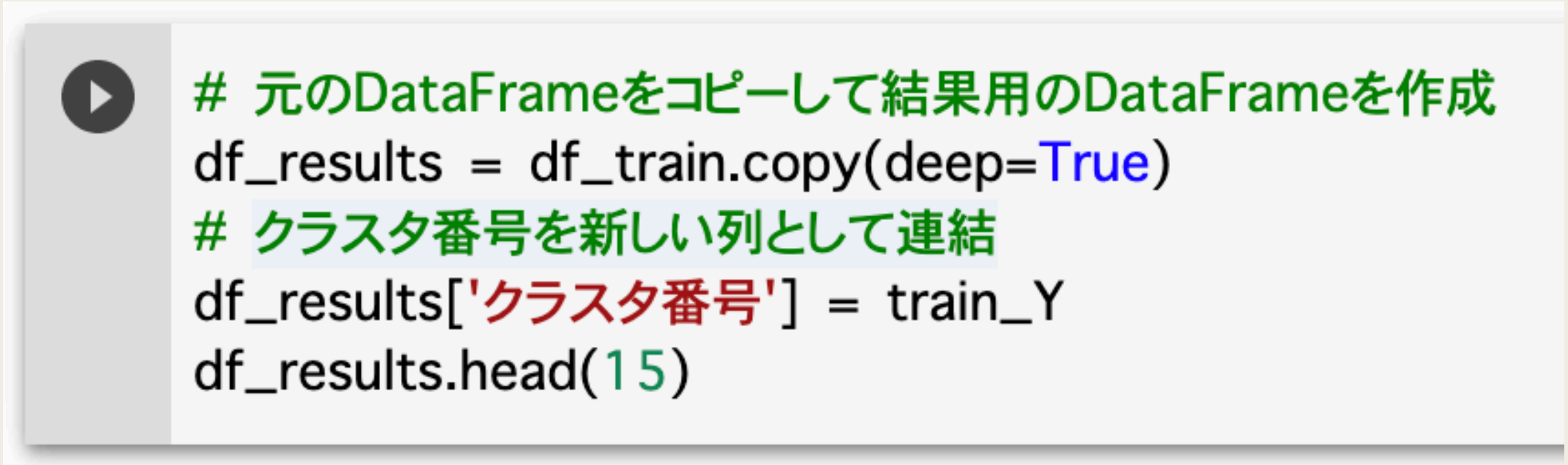
個数の確認

- 本当に４７個あるか、確認してみます。以下のコードで、47を表示されれば成功です。



クラスタ番号を、元のDataFrameに連結

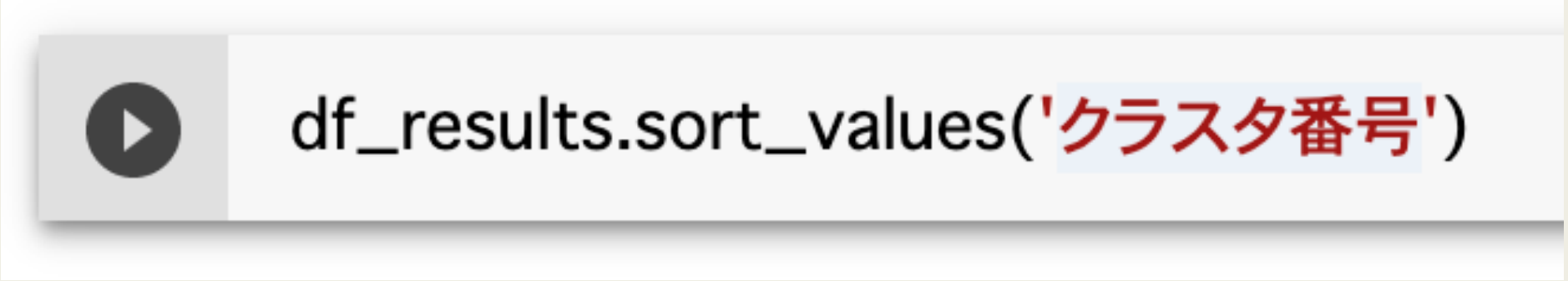
- 出力されたクラスタ番号を、元のDataFrameに新しい列として連結します。



```
# 元のDataFrameをコピーして結果用のDataFrameを作成
df_results = df_train.copy(deep=True)
# クラスタ番号を新しい列として連結
df_results['クラスタ番号'] = train_Y
df_results.head(15)
```

クラスタ番号の順に並び替え

- クラスタ番号順に整列し、表示してみましょう。



散布図で可視化

- クラスタ番号だけでは直感的な理解がしにくいのは確かです。そこで再度、散布図で可視化をします。今度はクラスタ番号ごとにプロットの色を変化させるこ とで見分けられるようにします。matplotlibのscatterで、オプション c= に色番号を指定することでプロットの色を変えられます。

```
# プロットの準備(x軸:食料、y軸:水道光熱費、色:クラスタ番号 )
plt.scatter(df_results['食料'], df_results['水道光熱費'], c=df_results['クラスタ番号'])
# X軸、Y軸のラベルを設定
plt.xlabel('Shokuryo')
plt.ylabel('Suido_Konetsu')
```

カラーコードを設定し、もう一度可視化

- 一部、見えにくい色がありますね。また、このままでは、どの色がどのクラスタ番号かが分かりにくいですので、カラーコードを管理するリストを作成して、もう一 度プロットしてみましょう。

```
# プロット/グラフ用のカラーリストを準備
LST_COLOR = ['r','g','b','c','m','y','k','lightping','goldenrod']
# プロットの準備(x軸:食料、y軸:水道光熱費、色:カラーリストのクラスタ番号 )
for i in range(len(df_results)):
  plt.scatter(df_results.iloc[i,1],df_results.iloc[i,2],c=LST_COLOR[df_results.iloc[i,3]],label=df_results.iloc[i,3] )
# X軸、Y軸のラベルを設定
plt.xlabel('Shokuryo')
plt.ylabel('Suido_Konetsu')
```

- **補足:** カラーリストは、

  0...赤、1...緑、2...青、3...シアン、4...マゼンダ、5...黄、、、となっています。

適切なクラスタ数

- 以上で、都道府県のクラスタリングが完了しました。

しかし、クラスタ数は、本当６個でよかったのでしょうか？６個と言うのは、適当に決めた値です。

- た完璧な方法ではありませんが、このクラスタ数を推定するための手法として「エルボー法」や「シルエット分析」という手法があります。ここでは比較的分かりやすい「エルボー法」についてトライしてみましょう。

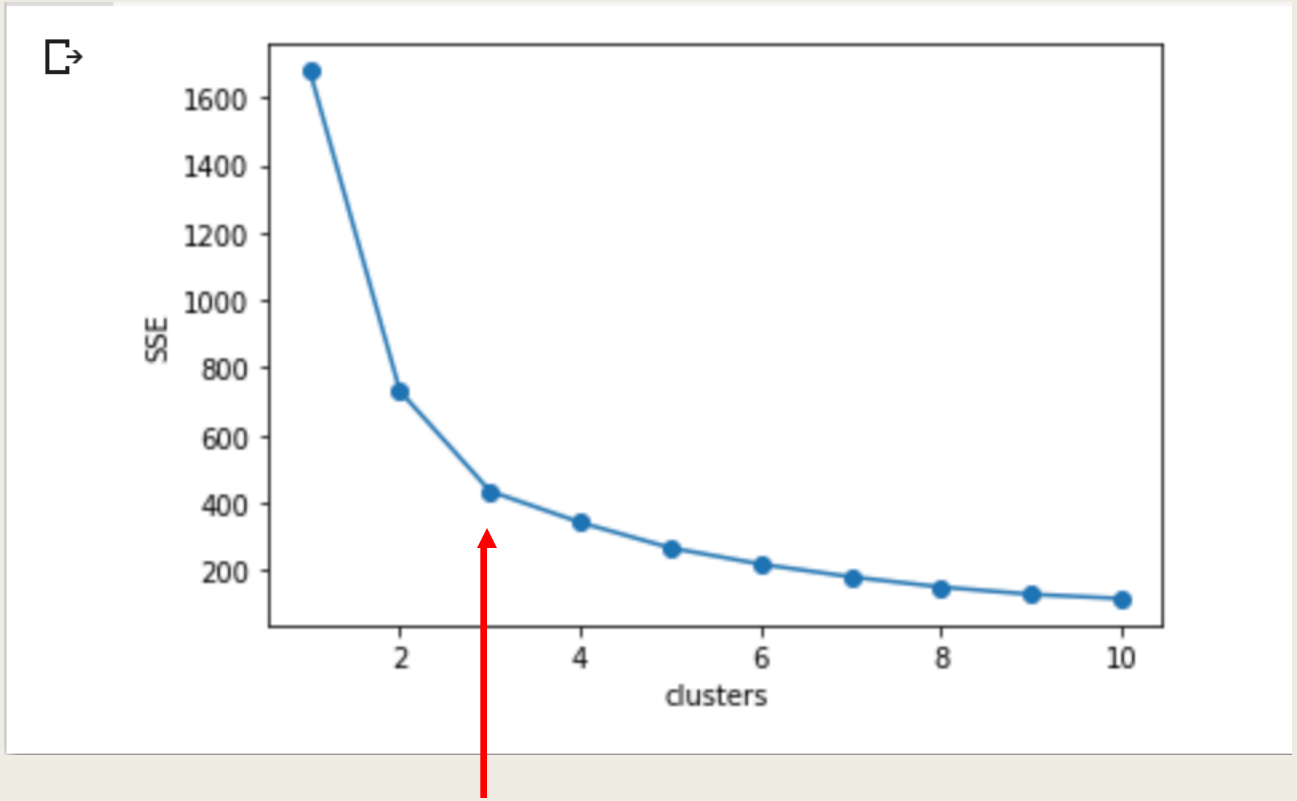
エルボー法で、適切なクラスタ数を算出

- k平均法のアルゴリズムを実行(fit)すると、各クラスタの重心との距離(残差二乗和)を取得することができます。そこでクラスタ数を総当り的に変化させながら、この 距離をプロットすることで、適切なクラスタ数を見つけ出そうというのが、エルボー法になります。

```
# k平均法(k-means)のライブラリをインポート
from sklearn.cluster import KMeans

lst_history = []

for i in range(1,11): # 1~10クラスタまで一気に計算
  # k-meansのモデルを生成
  kmeans = KMeans(n_clusters=i,random_state=0)
  kmeans.fit(train_X)    # クラスタリングを実行
  lst_history.append(kmeans.inertia_) # inertia_=残差平方和を得る

# 残差二乗和をプロット
plt.plot(range(1,11),lst_history,marker='o')
plt.xlabel('clusters') #クラスタ数
plt.ylabel('SSE')    #残差二乗和
plt.show()
```

実行結果を確認

- 横軸にクラスタ数、縦軸に残差平方和を図示すると、肘(エルボー)のようにポキっと曲がった場所があることがわかります。 上記の例ではクラスタ数=3の点でグ ラフがポキっと曲がっていますね。 エルボー法ではこのような点を見つけることで最適な(と思われる)クラスタ数を推定します。



■最後に、適切なクラスタ数（３）で再度クラスタリングを行ってみよう。