



# 技能向上訓練 A I 基礎コース

Python演習①  
クラスタリング

# データのダウンロード

- [https://github.com/Glbson5527/AI\\_ensyu](https://github.com/Glbson5527/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

# 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。



```
from google.colab import drive  
drive.mount('/content/drive')
```

- 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。

... Go to this URL in a browser: <https://accounts.google.com/o/oauth2/auth>

Enter your authorization code:

# 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_
```

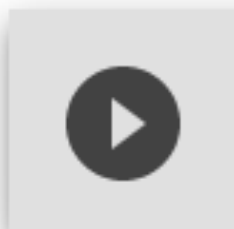
```
print(train_Y)
```



```
[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 4 7 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。



```
train_Y.shape
```

# 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。



```
df_results.sort_values('クラスタ番号')
```

# 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番号かが分かりにくいので、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','lightpink','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

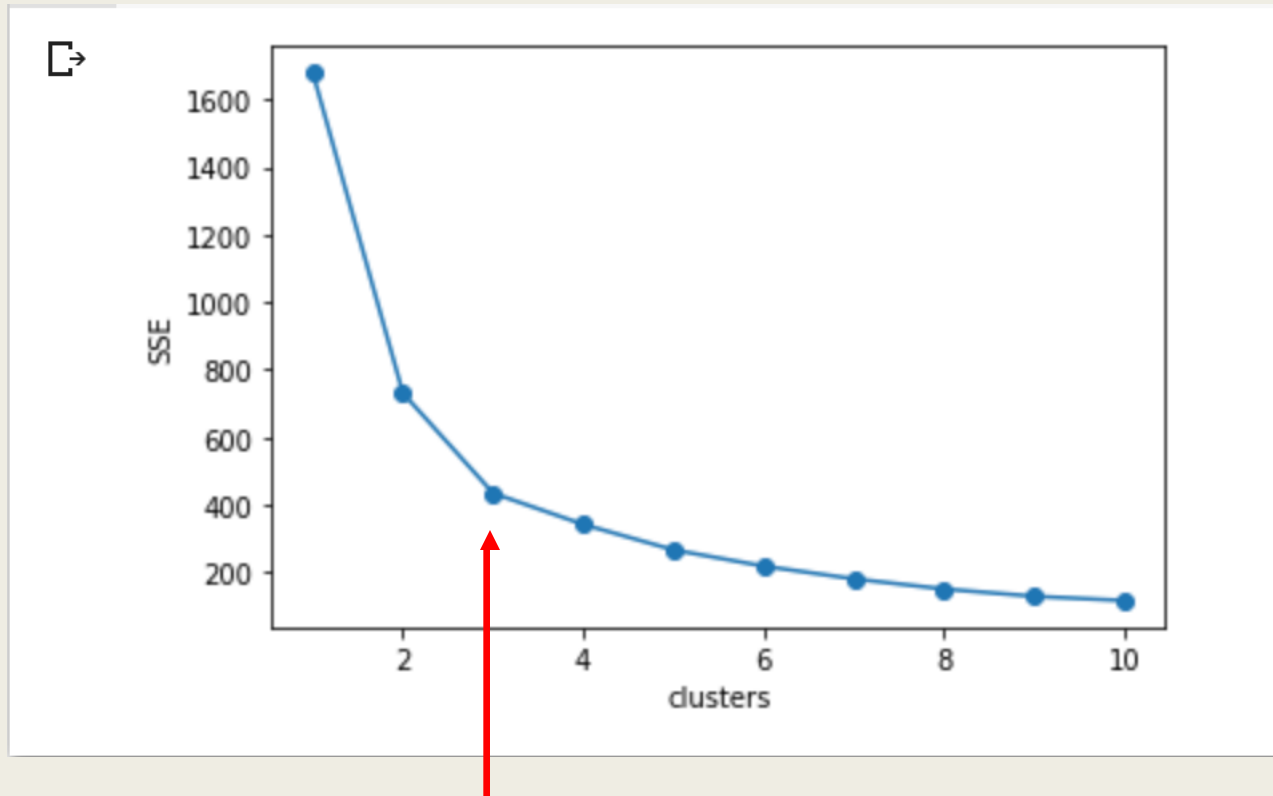
lst_history = []

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)のようにポキッ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ッ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。