fuzzy-c-mean

December 18, 2023

[55]:

```
!pip install fuzzy_c_means
```

```
Requirement already satisfied: fuzzy_c_means in /opt/conda/lib/python3.10/site-
packages (1.7.0)
Requirement already satisfied: joblib<2.0.0,>=1.2.0 in
/opt/conda/lib/python3.10/site-packages (from fuzzy_c_means) (1.3.2)
Requirement already satisfied: numpy<2.0.0,>=1.21.1 in
/opt/conda/lib/python3.10/site-packages (from fuzzy_c_means) (1.24.3)
Requirement already satisfied: pydantic<2.0.0,>=1.9.0 in
/opt/conda/lib/python3.10/site-packages (from fuzzy_c_means) (1.10.12)
Requirement already satisfied: tabulate<0.9.0,>=0.8.9 in
/opt/conda/lib/python3.10/site-packages (from fuzzy_c_means) (0.8.10)
Requirement already satisfied: tqdm<5.0.0,>=4.64.1 in
/opt/conda/lib/python3.10/site-packages (from fuzzy_c_means) (4.66.1)
Requirement already satisfied: typer<0.5.0,>=0.4.0 in
/opt/conda/lib/python3.10/site-packages (from fuzzy_c_means) (0.4.2)
Requirement already satisfied: typing-extensions>=4.2.0 in
/opt/conda/lib/python3.10/site-packages (from
pydantic<2.0.0,>=1.9.0->fuzzy_c_means) (4.5.0)
Requirement already satisfied: click<9.0.0,>=7.1.1 in
/opt/conda/lib/python3.10/site-packages (from
typer<0.5.0,>=0.4.0->fuzzy_c_means) (8.1.7)
```

[56]:

```
import numpy as np
import pandas as pd
import matplotlib .pyplot as plt
import seaborn as sns

import plotly.express as px
import plotly.figure_factory as ff

from fcmeans import FCM
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder
```

[57]:

```
data = pd.read_csv('/kaggle/input/unsupervised-learning-on-country-data/
 ↪Country-data.csv')
data.head()
```

[57]:
```
              country  child_mort  exports  health  imports  income  \
0          Afghanistan        90.2     10.0    7.58     44.9    1610
1              Albania        16.6     28.0    6.55     48.6    9930
2              Algeria        27.3     38.4    4.17     31.4   12900
3               Angola       119.0     62.3    2.85     42.9    5900
4  Antigua and Barbuda        10.3     45.5    6.03     58.9   19100

   inflation  life_expec  total_fer   gdpp
0       9.44        56.2       5.82    553
1       4.49        76.3       1.65   4090
2      16.10        76.5       2.89   4460
3      22.40        60.1       6.16   3530
4       1.44        76.8       2.13  12200
```

[58]:
```python
data.shape
```

[58]:
```
(167, 10)
```

[59]:
```python
data.columns
```

[59]:
```
Index(['country', 'child_mort', 'exports', 'health', 'imports', 'income',
       'inflation', 'life_expec', 'total_fer', 'gdpp'],
      dtype='object')
```

[60]:
```python
columns = data.columns
columns
```

[60]:
```
Index(['country', 'child_mort', 'exports', 'health', 'imports', 'income',
       'inflation', 'life_expec', 'total_fer', 'gdpp'],
      dtype='object')
```

[61]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   country     167 non-null    object
 1   child_mort  167 non-null    float64
 2   exports     167 non-null    float64
 3   health      167 non-null    float64
 4   imports     167 non-null    float64
 5   income      167 non-null    int64
 6   inflation   167 non-null    float64
 7   life_expec  167 non-null    float64
 8   total_fer   167 non-null    float64
 9   gdpp        167 non-null    int64
```

```
dtypes: float64(7), int64(2), object(1)
memory usage: 13.2+ KB
```

[62]:
```
data.describe()
```

[62]:
```
        child_mort     exports      health     imports         income  \
count   167.000000  167.000000  167.000000  167.000000     167.000000
mean     38.270060   41.108976    6.815689   46.890215   17144.688623
std      40.328931   27.412010    2.746837   24.209589   19278.067698
min       2.600000    0.109000    1.810000    0.065900     609.000000
25%       8.250000   23.800000    4.920000   30.200000    3355.000000
50%      19.300000   35.000000    6.320000   43.300000    9960.000000
75%      62.100000   51.350000    8.600000   58.750000   22800.000000
max     208.000000  200.000000   17.900000  174.000000  125000.000000

         inflation  life_expec   total_fer           gdpp
count   167.000000  167.000000  167.000000     167.000000
mean      7.781832   70.555689    2.947964   12964.155689
std      10.570704    8.893172    1.513848   18328.704809
min      -4.210000   32.100000    1.150000     231.000000
25%       1.810000   65.300000    1.795000    1330.000000
50%       5.390000   73.100000    2.410000    4660.000000
75%      10.750000   76.800000    3.880000   14050.000000
max     104.000000   82.800000    7.490000  105000.000000
```

Label Encoding Convert data from string and number

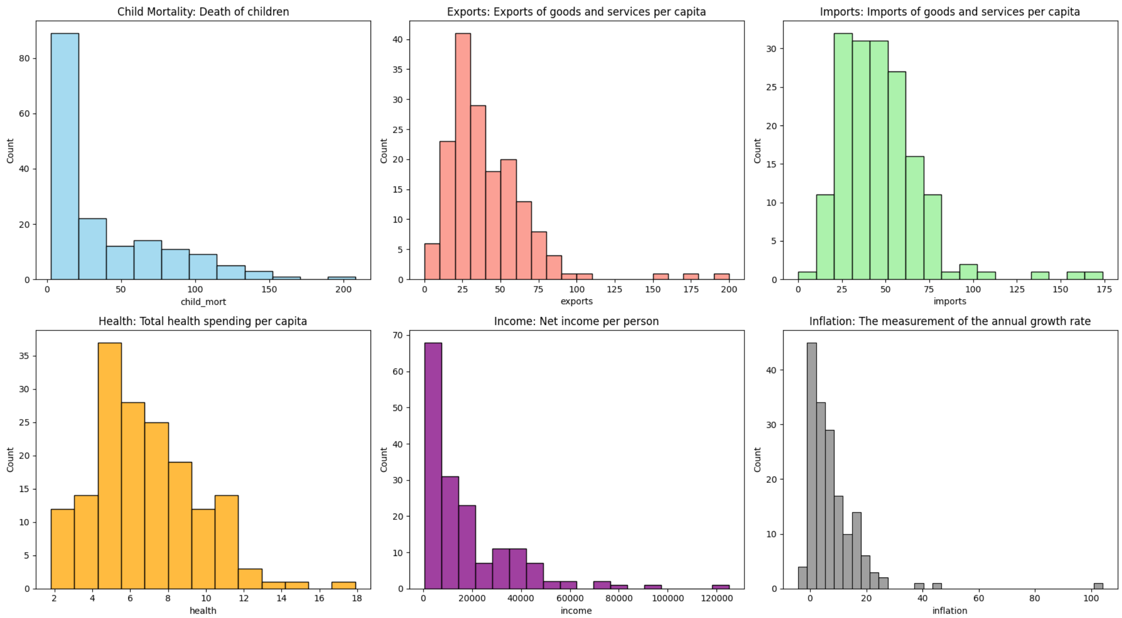
[63]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le.fit(data['country'])
le.transform(data['country'])
data['country'] = le.transform(data['country'])
```

[64]:
```
data.head()
```

[64]:
```
   country  child_mort  exports  health  imports  income  inflation  \
0        0        90.2     10.0    7.58     44.9    1610       9.44
1        1        16.6     28.0    6.55     48.6    9930       4.49
2        2        27.3     38.4    4.17     31.4   12900      16.10
3        3       119.0     62.3    2.85     42.9    5900      22.40
4        4        10.3     45.5    6.03     58.9   19100       1.44

   life_expec  total_fer   gdpp
0        56.2       5.82    553
1        76.3       1.65   4090
2        76.5       2.89   4460
3        60.1       6.16   3530
4        76.8       2.13  12200
```

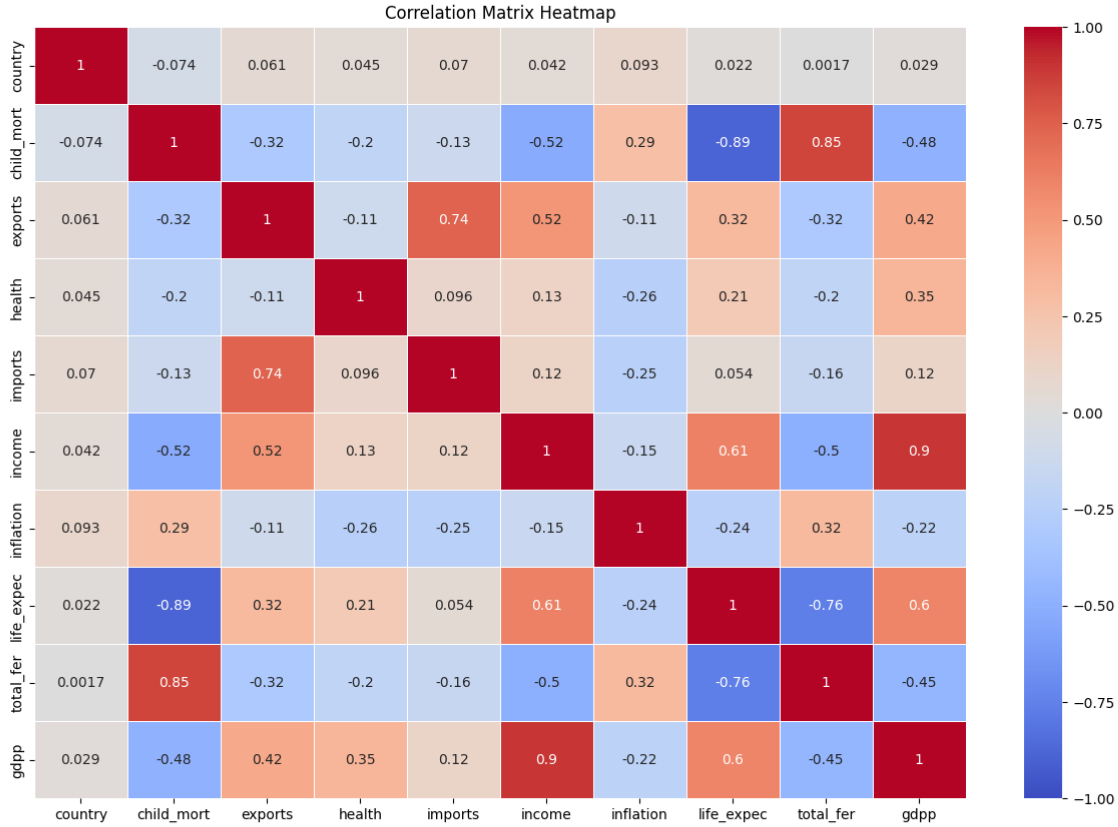
[65]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Create subplots with 2 rows and 3 columns
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(18, 10))

# Plot Child Mortality
sns.histplot(data["child_mort"], ax=axes[0, 0], color='skyblue')
axes[0, 0].set_title("Child Mortality: Death of children")

# Plot Exports
sns.histplot(data["exports"], ax=axes[0, 1], color='salmon')
axes[0, 1].set_title("Exports: Exports of goods and services per capita")

# Plot Imports
sns.histplot(data["imports"], ax=axes[0, 2], color='lightgreen')
axes[0, 2].set_title("Imports: Imports of goods and services per capita")

# Plot Health
sns.histplot(data["health"], ax=axes[1, 0], color='orange')
axes[1, 0].set_title("Health: Total health spending per capita")

# Plot Income
sns.histplot(data["income"], ax=axes[1, 1], color='purple')
axes[1, 1].set_title("Income: Net income per person")

# Plot Inflation
sns.histplot(data["inflation"], ax=axes[1, 2], color='gray')
axes[1, 2].set_title("Inflation: The measurement of the annual growth rate")

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
```



[66]:

```
plt.figure(figsize=(15, 10))
corr_matrix = data.corr()
# Plot the heatmap with annotations
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", vmin=-1, vmax=1,␣
 ↪center=0, linewidths=.5)
plt.title("Correlation Matrix Heatmap")
plt.show()
```



Scaling Data

[67]:
```
scalarModel = StandardScaler()
data = scalarModel.fit_transform(data)
data
```

[67]:
```
array([[-1.72171011,  1.29153238, -1.13827979, …, -1.61909203,
         1.90288227, -0.67917961],
       [-1.70096662, -0.5389489 , -0.47965843, …,  0.64786643,
        -0.85997281, -0.48562324],
       [-1.68022312, -0.27283273, -0.09912164, …,  0.67042323,
        -0.0384044 , -0.46537561],
       …,
       [ 1.68022312, -0.37231541,  1.13030491, …,  0.28695762,
        -0.66120626, -0.63775406],
       [ 1.70096662,  0.44841668, -0.40647827, …, -0.34463279,
         1.14094382, -0.63775406],
       [ 1.72171011,  1.11495062, -0.15034774, …, -2.09278484,
         1.6246091 , -0.62954556]])
```

Apply Fuzzy C Means Algorithm

[68]:
```
fcmModel = FCM(n_clusters = 4)
fcmModel.fit(data)
center = fcmModel.centers
center
```

[68]:
```
array([[-0.09169945, -0.70946541,  0.14862768,  0.8339647 , -0.15228512,
         0.9696234 , -0.405455  ,  0.92309281, -0.66655248,  1.24445092],
       [ 0.05514246, -0.28018828,  0.09816305, -0.13709174,  0.12265911,
        -0.08856392, -0.03198531,  0.17430523, -0.2971879 , -0.18351043],
       [-0.00458732, -0.23164428,  0.02566216, -0.17516046,  0.04819663,
        -0.12641153,  0.01104886,  0.13185398, -0.24482182, -0.21366038],
       [-0.10205958,  1.25691318, -0.43059624, -0.30416946, -0.21111204,
        -0.65948176,  0.17926978, -1.14365391,  1.30708304, -0.56897737]])
```

[69]:
```
#Calculating Prediction
pred = fcmModel.predict(data)
print('Predicted Value for fcmModel is : ' , pred)
pred.shape
```

```
Predicted Value for fcmModel is :  [3 2 2 3 1 2 2 0 0 2 0 0 2 1 2 0 2 3 2 2 1 2
2 0 1 3 3 2 3 0 2 3 3 2 2 2 3
 3 3 0 3 1 0 0 0 2 2 2 2 3 3 1 2 0 0 3 3 1 0 3 0 2 2 3 3 1 3 1 0 2 2 2 2 0
 0 0 2 0 1 2 3 3 0 1 3 1 1 3 3 1 1 0 1 3 3 1 1 3 0 3 1 1 1 2 1 2 3 3 3 2 0
 0 3 3 0 1 3 1 1 2 2 1 0 0 1 2 3 2 1 3 1 1 3 0 1 0 1 3 0 0 2 1 3 1 0 0 2 3
 1 3 3 2 1 1 1 3 1 0 0 0 1 2 1 2 1 3 3]
```

[69]:
```
(167,)
```

[70]:
```
data = pd.DataFrame(data , columns = columns )
data
```

[70]:
```
      country  child_mort   exports    health   imports    income  inflation  \
0   -1.721710    1.291532 -1.138280  0.279088 -0.082455 -0.808245   0.157336
1   -1.700967   -0.538949 -0.479658 -0.097016  0.070837 -0.375369  -0.312347
2   -1.680223   -0.272833 -0.099122 -0.966073 -0.641762 -0.220844   0.789274
3   -1.659480    2.007808  0.775381 -1.448071 -0.165315 -0.585043   1.387054
4   -1.638736   -0.695634  0.160668 -0.286894  0.497568  0.101732  -0.601749
..        …           …         …         …         …         …          …
162  1.638736   -0.225578  0.200917 -0.571711  0.240700 -0.738527  -0.489784
163  1.659480   -0.526514 -0.461363 -0.695862 -1.213499 -0.033542   3.616865
164  1.680223   -0.372315  1.130305  0.008877  1.380030 -0.658404   0.409732
165  1.700967    0.448417 -0.406478 -0.597272 -0.517472 -0.658924   1.500916
166  1.721710    1.114951 -0.150348 -0.338015 -0.662477 -0.721358   0.590015

     life_expec  total_fer      gdpp
0     -1.619092   1.902882 -0.679180
1      0.647866  -0.859973 -0.485623
```

```
2      0.670423  -0.038404 -0.465376
3     -1.179234   2.128151 -0.516268
4      0.704258  -0.541946 -0.041817
..          ...        ...       ...
162   -0.852161   0.365754 -0.546913
163    0.546361  -0.316678  0.029323
164    0.286958  -0.661206 -0.637754
165   -0.344633   1.140944 -0.637754
166   -2.092785   1.624609 -0.629546

[167 rows x 10 columns]
```

[71]:
```python
# add the cluster column to the dataframe
data['cluster'] = pred
data.head()
```

[71]:
```
    country  child_mort   exports    health   imports    income  inflation  \
0 -1.721710    1.291532 -1.138280  0.279088 -0.082455 -0.808245   0.157336
1 -1.700967   -0.538949 -0.479658 -0.097016  0.070837 -0.375369  -0.312347
2 -1.680223   -0.272833 -0.099122 -0.966073 -0.641762 -0.220844   0.789274
3 -1.659480    2.007808  0.775381 -1.448071 -0.165315 -0.585043   1.387054
4 -1.638736   -0.695634  0.160668 -0.286894  0.497568  0.101732  -0.601749

   life_expec  total_fer      gdpp  cluster
0   -1.619092   1.902882 -0.679180        3
1    0.647866  -0.859973 -0.485623        2
2    0.670423  -0.038404 -0.465376        2
3   -1.179234   2.128151 -0.516268        3
4    0.704258  -0.541946 -0.041817        1
```

[72]:
```python
# Visualizing the clusters
plt.scatter(data.loc[data['cluster'] == 0, 'child_mort'], data.
 ↪loc[data['cluster'] == 0, 'exports'], s=10, c='r', label='Cluster 0')
plt.scatter(data.loc[data['cluster'] == 1, 'child_mort'], data.
 ↪loc[data['cluster'] == 1, 'exports'], s=10, c='b', label='Cluster 1')
plt.scatter(data.loc[data['cluster'] == 2, 'child_mort'], data.
 ↪loc[data['cluster'] == 2, 'exports'], s=10, c='g', label='Cluster 2')
plt.scatter(data.loc[data['cluster'] == 3, 'child_mort'], data.
 ↪loc[data['cluster'] == 3, 'exports'], s=10, c='y', label='Cluster 3')

# Plotting cluster centers
plt.scatter(center[:, 0], center[:, 1], s=300, c='black', marker='+',␣
 ↪label='Cluster Centers')

plt.title('Clusters of countries')
plt.xlabel('Child Mortality')
plt.ylabel('Exports')
```

```
plt.legend()
plt.show()
```