In [16]:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.preprocessing import StandardScaler, normalize

from scipy.cluster.hierarchy import single, cophenet
from scipy.spatial.distance import pdist, squareform

from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score

from sklearn.cluster import KMeans
```

In [2]:

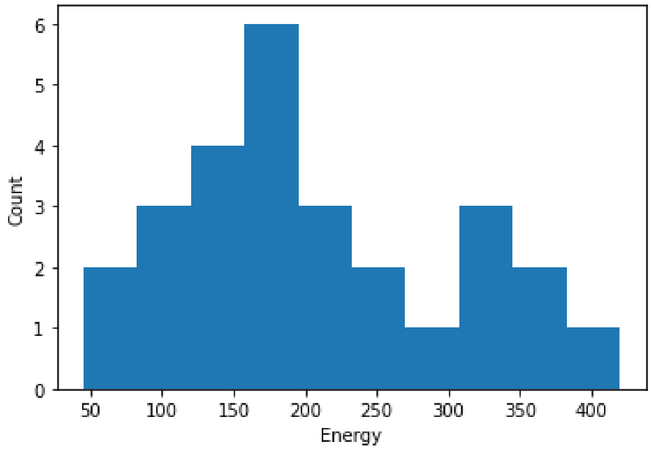
```
df = pd.read_csv('data.csv', names=['Name', 'Energy', 'Protein', 'Fat', 'Calcium', 'Iron'])
df
```

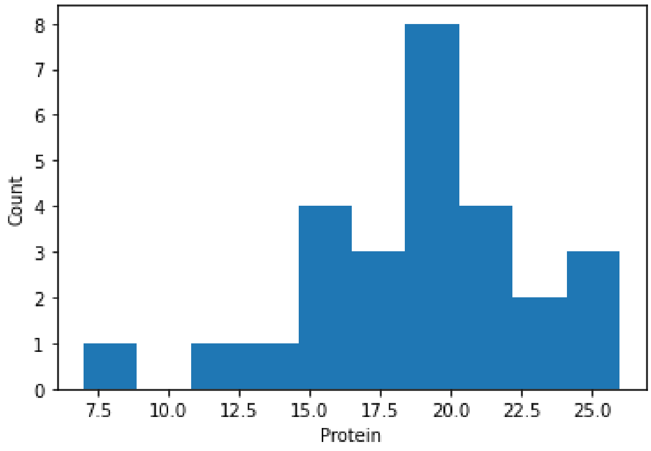
Out[2]:

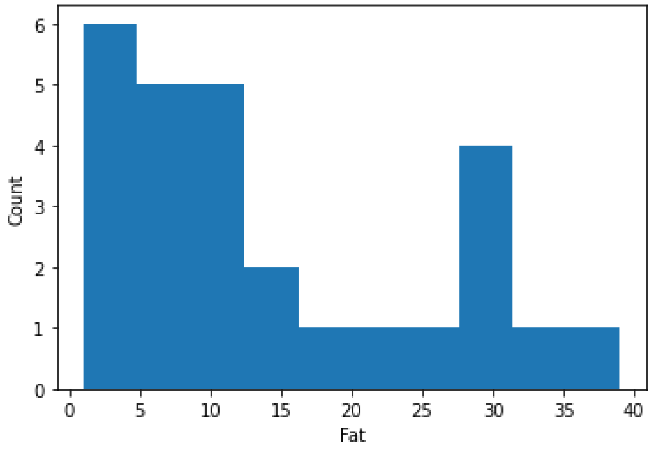
| | Name | Energy | Protein | Fat | Calcium | Iron |
|---|---|---|---|---|---|---|
| **0** | Braised beef | 340 | 20 | 28 | 9 | 2.6 |
| **1** | Hamburger | 245 | 21 | 17 | 9 | 2.7 |
| **2** | Roast beef | 420 | 15 | 39 | 7 | 2.0 |
| **3** | Beefsteak | 375 | 19 | 32 | 9 | 2.6 |
| **4** | Canned beef | 180 | 22 | 10 | 17 | 3.7 |
| **5** | Broiled chicken | 115 | 20 | 3 | 8 | 1.4 |
| **6** | Canned chicken | 170 | 25 | 7 | 12 | 1.5 |
| **7** | Beef heart | 160 | 26 | 5 | 14 | 5.9 |
| **8** | Roast lamb leg | 265 | 20 | 20 | 9 | 2.6 |
| **9** | Roast lamb shoulder | 300 | 18 | 25 | 9 | 2.3 |
| **10** | Smoked ham | 340 | 20 | 28 | 9 | 2.5 |
| **11** | Pork roast | 340 | 19 | 29 | 9 | 2.5 |
| **12** | Pork simmered | 355 | 19 | 30 | 9 | 2.4 |
| **13** | Beef tongue | 205 | 18 | 14 | 7 | 2.5 |
| **14** | Veal cutlet | 185 | 23 | 9 | 9 | 2.7 |
| **15** | Baked bluefish | 135 | 22 | 4 | 25 | 0.6 |
| **16** | Raw clams | 70 | 11 | 1 | 82 | 6.0 |
| **17** | Canned clams | 45 | 7 | 1 | 74 | 5.4 |
| **18** | Canned crabmeat | 90 | 14 | 2 | 38 | 0.8 |
| **19** | Fried haddock | 135 | 16 | 5 | 15 | 0.5 |
| **20** | Broiled mackerel | 200 | 19 | 13 | 5 | 1.0 |
| **21** | Canned mackerel | 155 | 16 | 9 | 157 | 1.8 |
| **22** | Fried perch | 195 | 16 | 11 | 14 | 1.3 |
| **23** | Canned salmon | 120 | 17 | 5 | 159 | 0.7 |
| **24** | Canned sardines | 180 | 22 | 9 | 367 | 2.5 |
| **25** | Canned tuna | 170 | 25 | 7 | 7 | 1.2 |
| **26** | Canned shrimp | 110 | 23 | 1 | 98 | 2.6 |

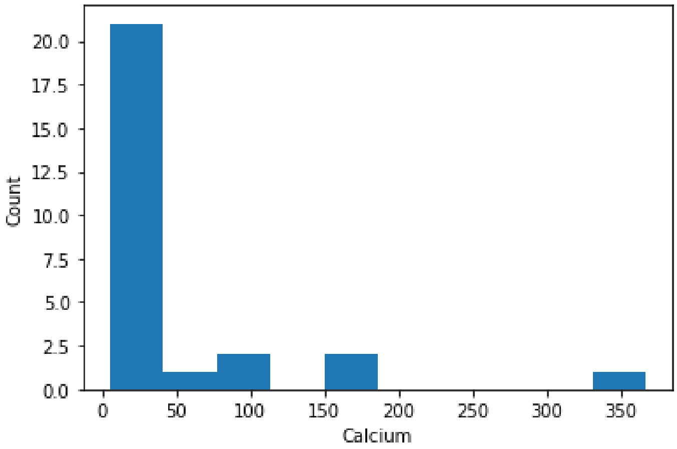
In [3]:

```
for i in range(1,6):
    plt.figure(figsize=(6,4))
    plt.hist(df.iloc[:,i].values)
    plt.xlabel(df.columns.values[i])
    plt.ylabel('Count')
    plt.show()
```

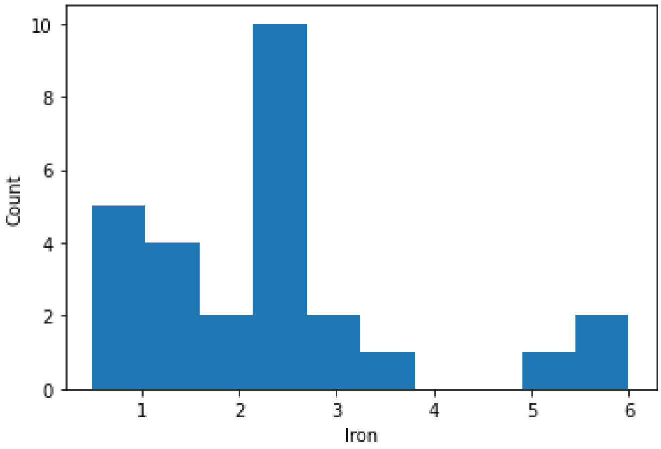








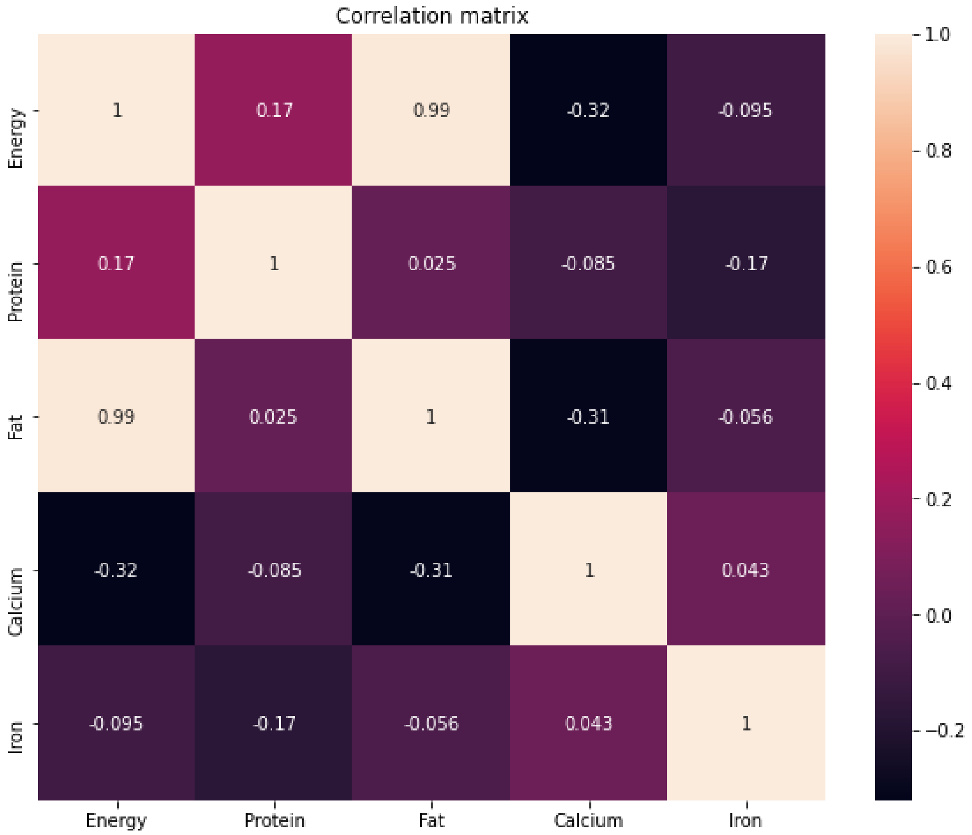
Count
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
50
100
150
200
250
300
350
Calcium



Count
10
8
6
4
2
0
1
2
3
4
5
6
Iron

In [4]:

```
plt.figure(figsize=(10,8))
g = sns.heatmap(df.corr(), annot=True)
g.set(title='Correlation matrix')
plt.show()
```



In [5]:

```
df2 = df.copy()
df = normalize(df.iloc[:,1:])
df
```

Out[5]:

```
array([[0.99454428, 0.0585026 , 0.08190365, 0.02632617, 0.00760534],
       [0.99325267, 0.08513594, 0.06891957, 0.03648683, 0.01094605],
       [0.99493947, 0.03553355, 0.09238724, 0.01658232, 0.00473781],
       [0.9948043 , 0.05040342, 0.08488997, 0.0238753 , 0.00689731],
       [0.98659111, 0.12058336, 0.05481062, 0.09317805, 0.02027993],
       [0.98251274, 0.17087178, 0.02563077, 0.06834871, 0.01196102],
       [0.98610394, 0.14501528, 0.04060428, 0.06960734, 0.00870092],
       [0.98228281, 0.15962096, 0.03069634, 0.08594975, 0.03622168],
       [0.99373848, 0.07499913, 0.07499913, 0.03374961, 0.00974989],
       [0.99429754, 0.05965785, 0.08285813, 0.02982893, 0.00762295],
       [0.99454645, 0.05850273, 0.08190383, 0.02632623, 0.00731284],
       [0.99446987, 0.05557332, 0.08482243, 0.0263242 , 0.00731228],
       [0.99469532, 0.05323721, 0.08405876, 0.02521763, 0.0067247 ],
       [0.99322533, 0.08721003, 0.06783002, 0.03391501, 0.0121125 ],
       [0.98995204, 0.12307512, 0.04815983, 0.04815983, 0.01444795],
       [0.97048593, 0.15815326, 0.02875514, 0.17971962, 0.00431327],
       [0.64489429, 0.10134053, 0.00921278, 0.7554476 , 0.05527665],
       [0.51686048, 0.08040052, 0.01148579, 0.84994835, 0.06202326],
       [0.91171607, 0.1418225 , 0.02026036, 0.38494679, 0.00810414],
       [0.98639392, 0.11690595, 0.03653311, 0.10959932, 0.00365331],
       [0.99312162, 0.09434655, 0.06455291, 0.02482804, 0.00496561],
       [0.7001169 , 0.07227013, 0.04065195, 0.70915067, 0.00813039],
       [0.99252785, 0.08143818, 0.05598875, 0.07125841, 0.00661685],
       [0.60003383, 0.08500479, 0.02500141, 0.79504482, 0.0035002 ],
       [0.43959976, 0.05372886, 0.02197999, 0.89629507, 0.00610555],
       [0.98769725, 0.1452496 , 0.04066989, 0.04066989, 0.00697198],
       [0.73759485, 0.15422438, 0.00670541, 0.65712995, 0.01743406]])
```

In [6]:

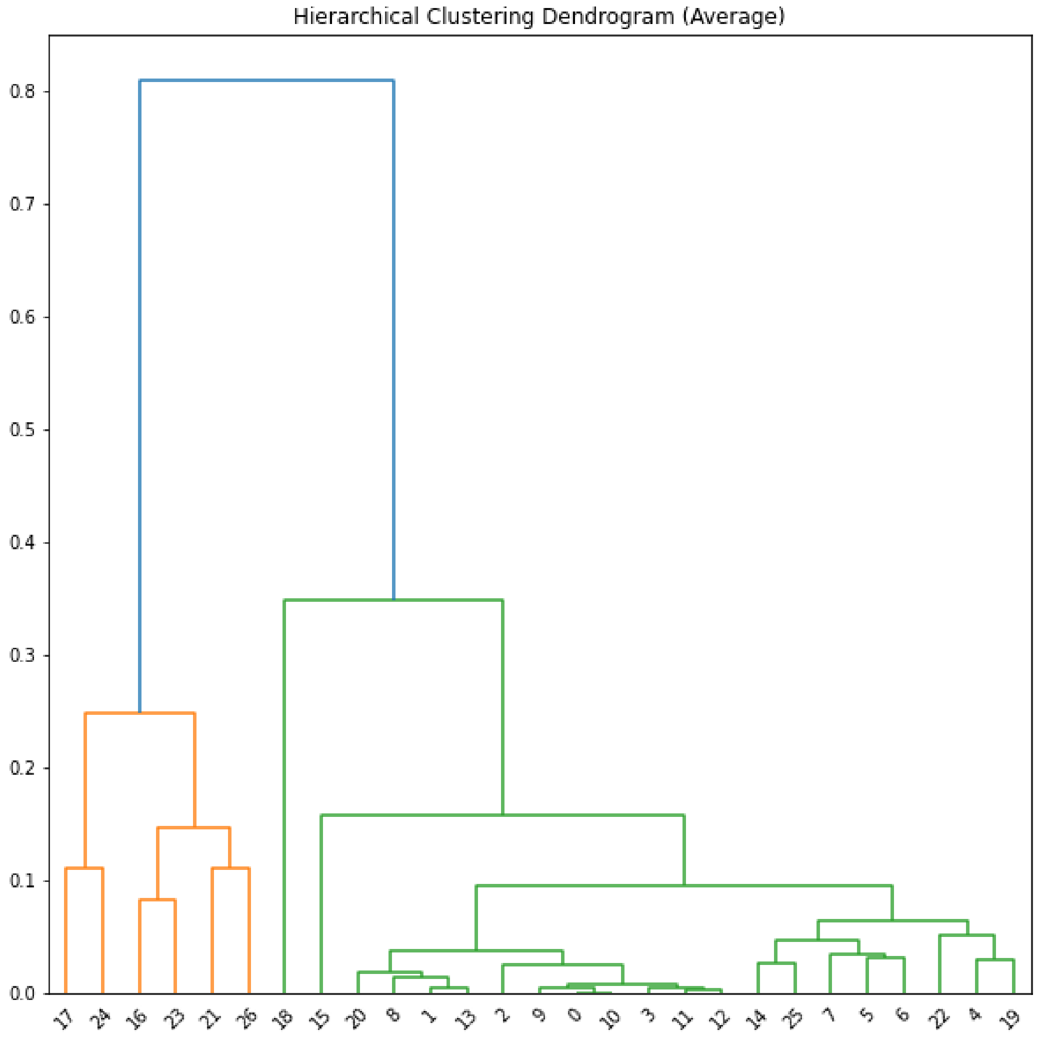
```
c1,coph=cophenet(linkage(df, 'single'),pdist(df))
c2,coph=cophenet(linkage(df, 'complete'),pdist(df))
c3,coph=cophenet(linkage(df, 'average'),pdist(df))
c4,coph=cophenet(linkage(df, 'weighted'),pdist(df))
c5,coph=cophenet(linkage(df, 'centroid'),pdist(df))
linkage=pd.DataFrame({"Linkage":['Single','Complete','Average','Weighted','Centroid'],
                    "Cophenet Coeff":[c1,c2,c3,c4,c5]})
linkage
```

Out[6]:

| | Linkage | Cophenet Coeff |
|---|---|---|
| **0** | Single | 0.959796 |
| **1** | Complete | 0.964865 |
| **2** | Average | 0.968280 |
| **3** | Weighted | 0.967990 |
| **4** | Centroid | 0.968274 |

In [15]:

```
plt.figure(figsize=(10, 10))
dendrogram(linkage(df, 'average'))
plt.title('Hierarchical Clustering Dendrogram (Average)')
plt.show()
```



In [39]:
```
db_index=[]
for i in range(2,11):
    db_index.append(silhouette_score(df, AgglomerativeClustering(n_clusters=i,linkage='aver
db_index = pd.DataFrame({'K Values':['2','3','4','5','6','7','8','9','10'],
                         'Silhouette index':db_index})
pd.DataFrame(db_index)
```

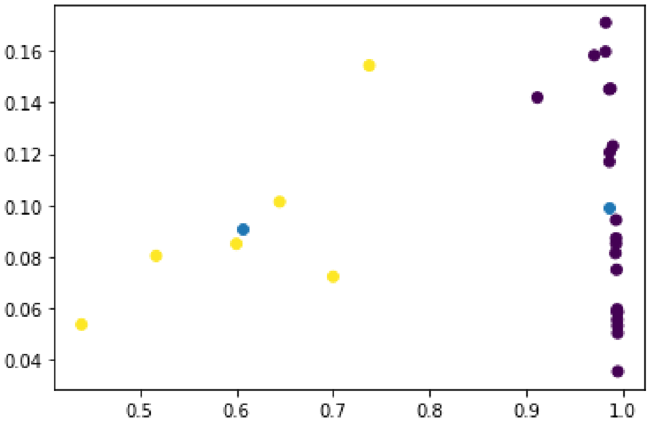
Out[39]:

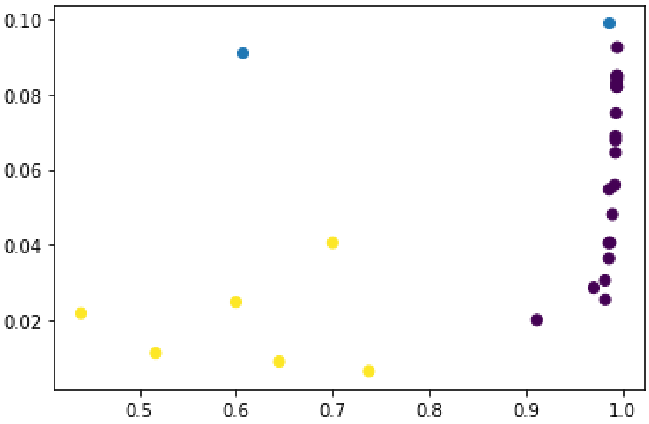
|  | K Values | Silhouette index |
|---|---|---|
| **0** | 2 | 0.840039 |
| **1** | 3 | 0.702438 |
| **2** | 4 | 0.673562 |
| **3** | 5 | 0.472026 |
| **4** | 6 | 0.441687 |
| **5** | 7 | 0.415750 |
| **6** | 8 | 0.385742 |
| **7** | 9 | 0.418950 |
| **8** | 10 | 0.402763 |

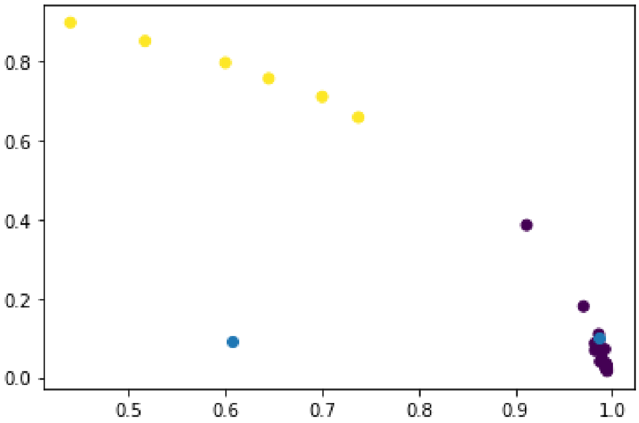
In [40]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(init="random", n_clusters=2)
kmeans.fit(df)
y_kmeans = kmeans.predict(df)
```

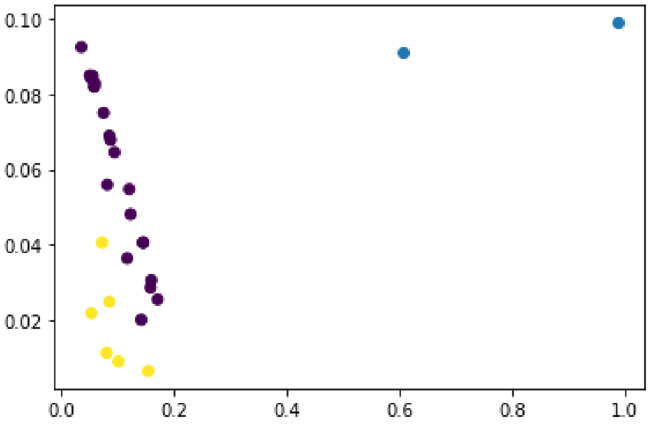
In [41]:

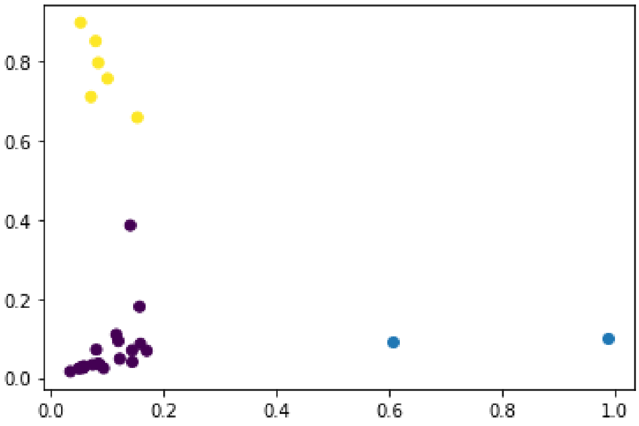
```
for i in range(4):
    for j in range(i+1, 4):
        plt.figure(figsize=(6, 4))
        plt.scatter(df[:, i], df[:, j], c=y_kmeans)

        centers = kmeans.cluster_centers_
        plt.scatter(centers[:, 0], centers[:, 1])
        plt.show()
```

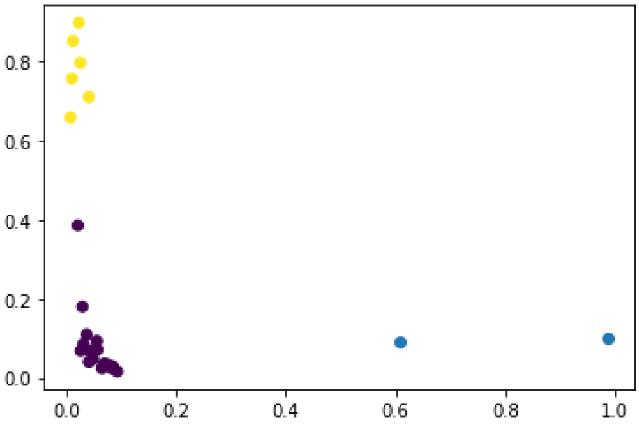







0.10
0.08
0.06
0.04
0.02
0.0
0.2
0.4
0.6
0.8
1.0


0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0