[WUM] Praca Domowa nr 7 Kacper Kurowski

June 15, 2021

1 [WUM] Praca Domowa nr7

1.1 Kacper Kurowski

Załadujmy wpierw zbiór danych

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import cluster, datasets, mixture
from sklearn import preprocessing
import seaborn as sns
import sklearn
```

[14]:
```
val = pd.read_csv( "val.csv")
train = pd.read_csv( "train.csv")
test = pd.read_csv( "test.csv")
train_vals = [0]*len( train)
```

i pobieżnie go obejrzmy

[79]:
```
val.head()
```

[79]:
```
   Alcohol  Malic acid   Ash  Alcalinity of ash  Magnesium  Total phenols  \
0    13.86        1.51  2.67               25.0         86           2.95
1    13.40        3.91  2.48               23.0        102           1.80
2    12.82        3.37  2.30               19.5         88           1.48
3    12.37        1.07  2.10               18.5         88           3.52
4    13.50        1.81  2.61               20.0         96           2.53

   Flavanoids  Nonflavanoid phenols  Proanthocyanins  Color intensity   Hue  \
0        2.86                  0.21             1.87             3.38  1.36
1        0.75                  0.43             1.41             7.30  0.70
2        0.66                  0.40             0.97            10.26  0.72
3        3.75                  0.24             1.95             4.50  1.04
4        2.61                  0.28             1.66             3.52  1.12

   OD280/OD315 of diluted wines  Proline
```

```
0                          3.16      410
1                          1.56      750
2                          1.75      685
3                          2.77      660
4                          3.82      845
```

[80]:
```
train.head()
```

[80]:
```
   Alcohol  Malic acid   Ash  Alcalinity of ash  Magnesium  Total phenols  \
0    12.72        1.75  2.28               22.5         84           1.38
1    13.23        3.30  2.28               18.5         98           1.80
2    12.58        1.29  2.10               20.0        103           1.48
3    12.37        1.17  1.92               19.6         78           2.11
4    13.84        4.12  2.38               19.5         89           1.80

   Flavanoids  Nonflavanoid phenols  Proanthocyanins  Color intensity   Hue  \
0        1.76                  0.48             1.63             3.30  0.88
1        0.83                  0.61             1.87            10.52  0.56
2        0.58                  0.53             1.40             7.60  0.58
3        2.00                  0.27             1.04             4.68  1.12
4        0.83                  0.48             1.56             9.01  0.57

   OD280/OD315 of diluted wines  Proline
0                          2.42      488
1                          1.51      675
2                          1.55      640
3                          3.48      510
4                          1.64      480
```

[82]:
```
test.head()
```

[82]:
```
   class  Alcohol  Malic acid   Ash  Alcalinity of ash  Magnesium  \
0      0    13.34        0.94  2.36               17.0        110
1      0    12.00        0.92  2.00               19.0         86
2      0    11.84        0.89  2.58               18.0         94
3      0    12.47        1.52  2.20               19.0        162
4      0    11.81        2.12  2.74               21.5        134

   Total phenols  Flavanoids  Nonflavanoid phenols  Proanthocyanins  \
0           2.53        1.30                  0.55             0.42
1           2.42        2.26                  0.30             1.43
2           2.20        2.21                  0.22             2.35
3           2.50        2.27                  0.32             3.28
4           1.60        0.99                  0.14             1.56

   Color intensity   Hue  OD280/OD315 of diluted wines  Proline
0             3.17  1.02                          1.93      750
```

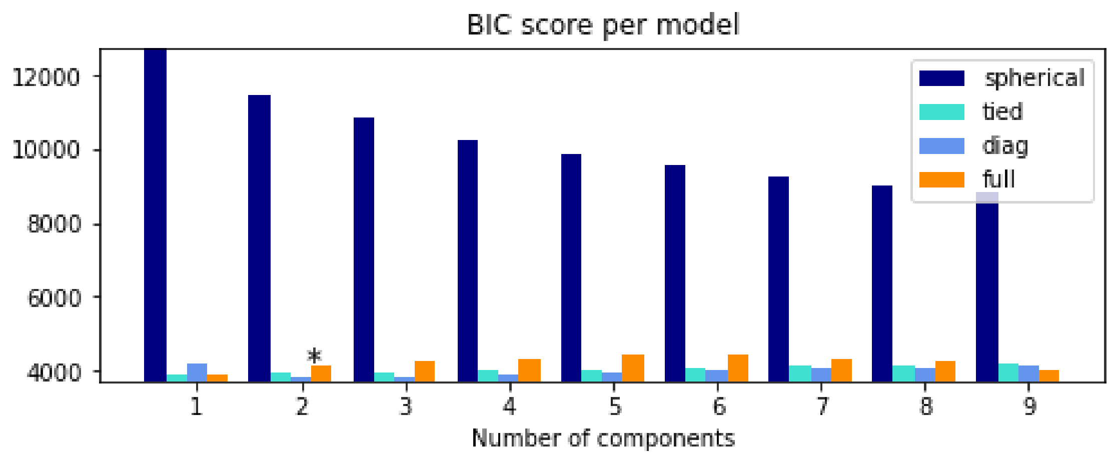
1	2.50	1.38	3.12	278
2	3.05	0.79	3.08	520
3	2.60	1.16	2.63	937
4	2.50	0.95	2.26	625

Postarajmy się znaleźć optymalne parametry dla GMM

[4]:
```
import numpy as np
import itertools

from scipy import linalg
import matplotlib.pyplot as plt
import matplotlib as mpl

from sklearn import mixture
```

[83]:
```
X = train

bics = [ [] for i in range( 30)]

lowest_bic = np.infty
bic = []
n_components_range = range(1, 10)
cv_types = ['spherical', 'tied', 'diag', 'full']
for cv_type in cv_types:
    for n_components in n_components_range:
        for t in range(30): # Powtórzmy eksperyment wielokrotnie, dla bardziej␣
 ↪miarodajnych wyników
            # Fit a Gaussian mixture with EM
            gmm = mixture.GaussianMixture(n_components=n_components,
                                          covariance_type=cv_type)
            gmm.fit(X, y = train_vals)
            bics[t] = gmm.bic(X)
        bic.append( np.mean( np.array( bics)))
        if bic[-1] < lowest_bic:
            lowest_bic = bic[-1]
            best_gmm = gmm

bic = np.array(bic)
color_iter = itertools.cycle(['navy', 'turquoise', 'cornflowerblue',
                              'darkorange'])
clf = best_gmm
bars = []

# Plot the BIC scores
plt.figure(figsize=(8, 6))
spl = plt.subplot(2, 1, 1)
```

```
for i, (cv_type, color) in enumerate(zip(cv_types, color_iter)):
    xpos = np.array(n_components_range) + .2 * (i - 2)
    bars.append(plt.bar(xpos, bic[i * len(n_components_range):
                                  (i + 1) * len(n_components_range)],
                        width=.2, color=color))
plt.xticks(n_components_range)
plt.ylim([bic.min() * 1.01 - .01 * bic.max(), bic.max()])
plt.title('BIC score per model')
xpos = np.mod(bic.argmin(), len(n_components_range)) + .65 +\
    .2 * np.floor(bic.argmin() / len(n_components_range))
plt.text(xpos, bic.min() * 0.97 + .03 * bic.max(), '*', fontsize=14)
spl.set_xlabel('Number of components')
spl.legend([b[0] for b in bars], cv_types)

plt.show()
```



W przypadku BIC score im mniejsza wartość, tym lepiej. Okazuje się, że najlepszy wynik dały 2 komponenty z diagonalną kowariancją

[71]:
```
gmm = mixture.GaussianMixture(
    n_components=2,
    covariance_type='diag'
).fit( train, y = train_vals)
```

[72]:
```
test_pred = gmm.predict( test.drop(['class'], axis=1))
test_true = test['class'].to_numpy()
```

[73]:
```
from sklearn.metrics import f1_score, precision_score, recall_score
```

[78]:
```
print( "f1_score: % 4.3f" % f1_score( test_true, test_hat))
print( "precision_score: % 4.3f" % precision_score( test_true, test_hat))
```

```
print( "recall_score: % 4.3f" % recall_score( test_true, test_hat))
```

```
f1_score:  0.556
precision_score:  0.385
recall_score:  1.000
```

Zatem, wyniki nie są bardzo wysokie, jednakże, na podstawie recall, wiemy, że gmm był w stanie znaleźć wszystkie outliery. Niestety, wiele inlierowych wartości zakwalifikował jako outliery, co wiemy chociażby na podstawie precision.