10_Clustering_Introduction

July 4, 2024

[1]:
```
from numpy import linspace, zeros, meshgrid, c_, logspace

from matplotlib.pyplot import subplots, show
from matplotlib.colors import LogNorm

from pandas import read_csv
```

1 Clustering

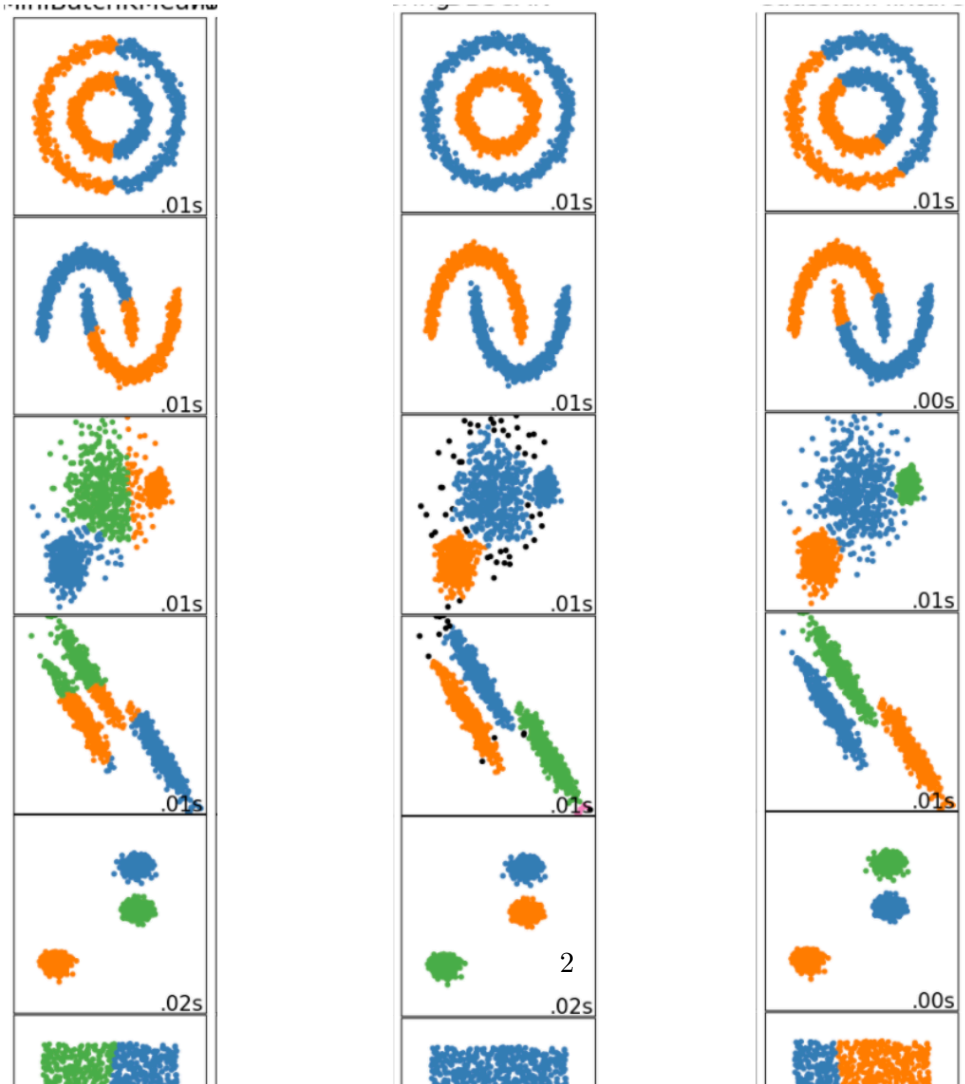
1.1 The Clustering Problem

If data are not labelled the

Clustering Methods

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

KMEANS DENSITY-BASED GAUSSIAN MIXTURE

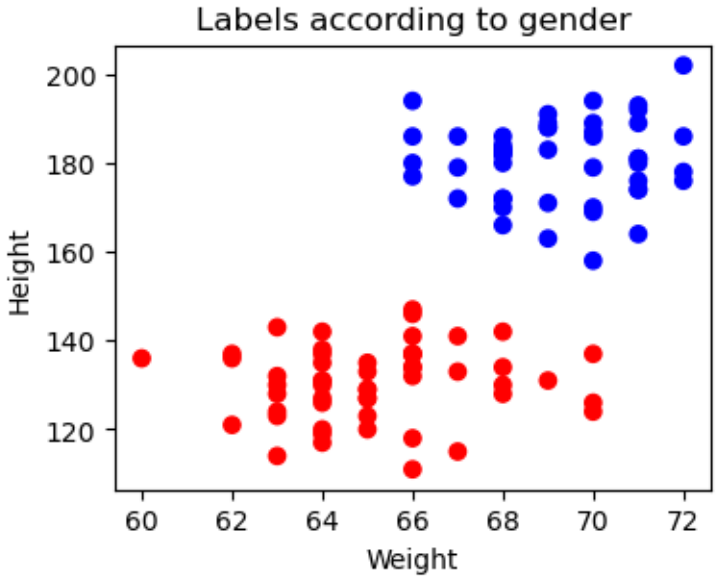


1.2 Weight-Height Data

[2]:
```
df = read_csv("../Data/patients_data.csv")

df.columns
```

[2]:
```
Index(['Age', 'Height', 'Weight', 'Systolic', 'Diastolic', 'Smoker', 'Gender'],
dtype='object')
```

[3]:
```
X = df.to_numpy()

X_sub_1 = X[:,1:3]


y_true = zeros(len(X[:,-1]))

y_true[X[:,-1] == 'Female'] = 1


fig, ax = subplots(figsize=(4, 3))

ax.scatter(X_sub_1[:, 0], X_sub_1[:, 1], c=y_true, cmap='bwr');
ax.set_title('Labels according to gender')
ax.set_xlabel('Weight')
ax.set_ylabel('Height');

show()
```



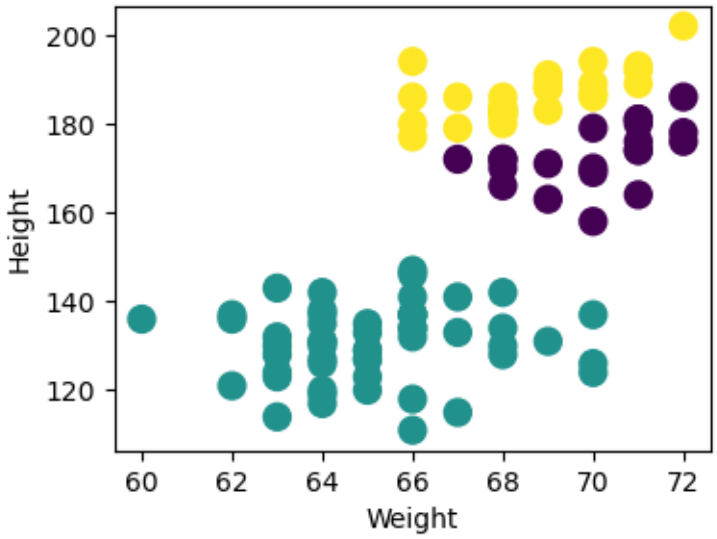
[ ]:

1.3 GMM Clustering

Optimises the likelihood that the data come from an overlap of (e.g. 2) normal distributions

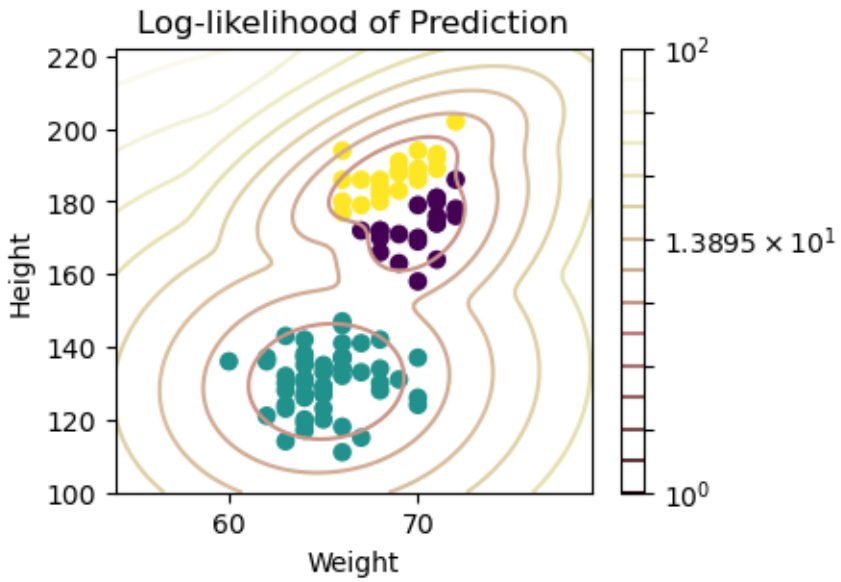
The method to opimise label distribution is Expectation maximisation

[5]:
```
from sklearn.mixture import GaussianMixture

n_components = 3

clu_1 = GaussianMixture(n_components=n_components)

clu_1.fit(X_sub_1)

y_predict_1 = clu_1.predict(X_sub_1)

fig, ax = subplots(figsize=(4, 3))

ax.scatter(X_sub_1[:, 0], X_sub_1[:, 1],  s=100, c=y_predict_1, cmap='viridis');
ax.set_xlabel('Weight')
ax.set_ylabel('Height');

show()
```



1.3.1 Log Likelihood

[11]:
```
resolution = 100

vec_a = linspace(0.9*min(X_sub_1[:,0]), 1.1*max(X_sub_1[:,0]), resolution)
vec_b = linspace(0.9*min(X_sub_1[:,1]), 1.1*max(X_sub_1[:,1]), resolution)

grid_a, grid_b = meshgrid(vec_a, vec_b)

XY_statespace = c_[grid_a.ravel(), grid_b.ravel()]

Z_score = clu_1.score_samples(XY_statespace)

Z_s = Z_score.reshape(grid_a.shape)


fig, ax = subplots(figsize=(4, 3))

cax = ax.contour(grid_a, grid_b, -Z_s,
           norm=LogNorm(vmin=1.0, vmax=100.0),
           levels=logspace(0, 2, 15),
           cmap='pink'
          )

fig.colorbar(cax)

ax.scatter(X_sub_1[:, 0], X_sub_1[:, 1],  c=y_predict_1, cmap='viridis')
ax.set_xlabel('Weight')
ax.set_ylabel('Height')

ax.set_title('Log-likelihood of Prediction', fontsize=12)

show()
```



1.3.2 Assessment

Similarity measure to compare the overlap of two clusterings

Assessment using extrenal standard done with rand_score

[19]:
```
from sklearn.metrics.cluster import adjusted_rand_score

scoring = adjusted_rand_score(y_true, y_predict_1)

print(scoring)
```

```
1.0
```

The scoring indicates perfect labelling according to 'Gender'.

1.3.3 Systolic-Diastolic Data

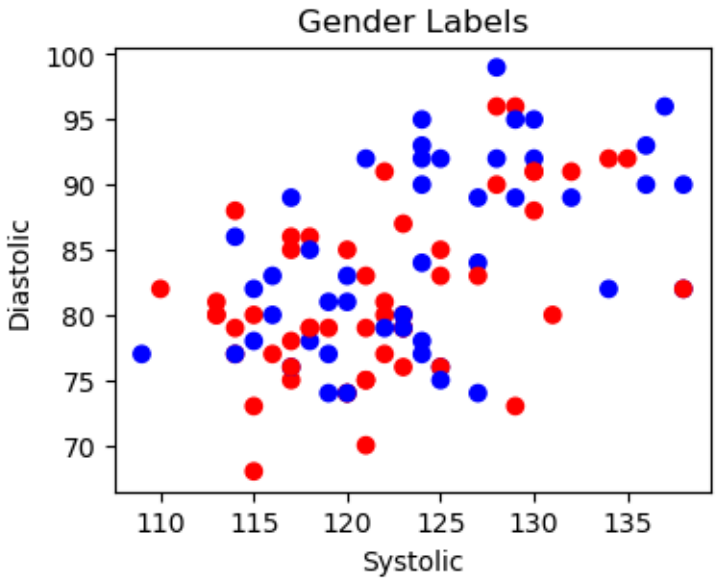
[31]:
```
df
```

[31]:

	Age	Height	Weight	Systolic	Diastolic	Smoker	Gender
0	38	71	176.0	124.0	93.0	1	Male
1	43	69	163.0	109.0	77.0	0	Male
2	38	64	131.0	125.0	83.0	0	Female
3	40	67	133.0	117.0	75.0	0	Female
4	49	64	119.0	122.0	80.0	0	Female
..	…	…	…	…	…	…	…
95	25	69	171.0	128.0	99.0	1	Male
96	44	69	188.0	124.0	92.0	1	Male

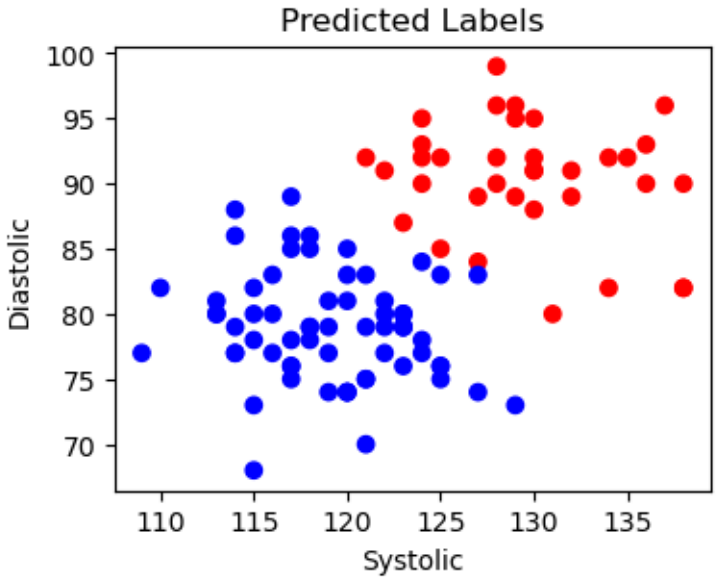
```
97   49      70   186.0     119.0       74.0       0    Male
98   45      68   172.0     136.0       93.0       1    Male
99   48      66   177.0     114.0       86.0       0    Male

[100 rows x 7 columns]
```

[12]:
```
X_sub_2 = X[:,3:5]

fig, ax = subplots(figsize=(4, 3))

ax.scatter(X_sub_2[:, 0], X_sub_2[:, 1], c=y_true, cmap='bwr');
ax.set_xlabel('Systolic')
ax.set_ylabel('Diastolic')
ax.set_title('Gender Labels');

show()
```



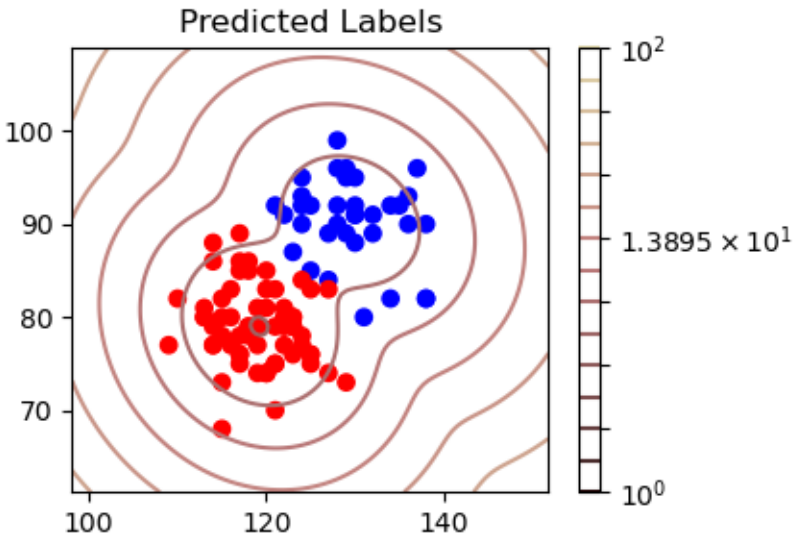
[13]:
```
n_components = 2

clu_2 = GaussianMixture(n_components=n_components)

clu_2.fit(X_sub_2)

y_predict_2 = clu_2.predict(X_sub_2)

fig, ax = subplots(figsize=(4, 3))
```

```
ax.scatter(X_sub_2[:, 0], X_sub_2[:, 1],  c=y_predict_2, cmap='bwr_r');
ax.set_title('Predicted Labels')
ax.set_xlabel('Systolic')
ax.set_ylabel('Diastolic');

show()
```



[17]:
```
resolution = 100

vec_a = linspace(0.9*min(X_sub_2[:,0]), 1.1*max(X_sub_2[:,0]), resolution)
vec_b = linspace(0.9*min(X_sub_2[:,1]), 1.1*max(X_sub_2[:,1]), resolution)

grid_a, grid_b = meshgrid(vec_a, vec_b)

XY_statespace = c_[grid_a.ravel(), grid_b.ravel()]

Z_score = clu_2.score_samples(XY_statespace)

Z_s = Z_score.reshape(grid_a.shape)


fig, ax = subplots(figsize=(4, 3))

cax = ax.contour(grid_a, grid_b, -Z_s,
           norm=LogNorm(vmin=1.0, vmax=1000.0),
           levels=logspace(0, 2, 15),
           cmap='pink'
```

```
          )

fig.colorbar(cax)

ax.scatter(X_sub_2[:, 0], X_sub_2[:, 1],  c=y_predict_2, cmap='bwr')

ax.set_title('Predicted Labels');

show()
```



[20]:
```
scoring = adjusted_rand_score(y_true, y_predict_2)

print(scoring)
```

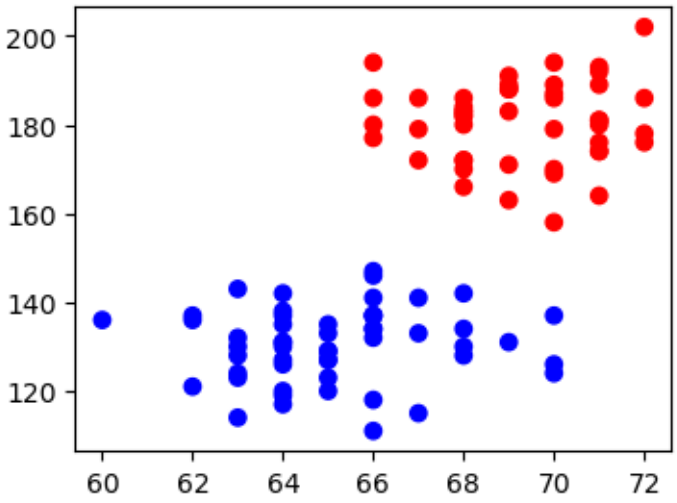
```
0.030809331315545804
```

The scoring indicates pure chance.

1.4 Kmeans Clustering

[21]:
```
from sklearn.cluster import KMeans

n_components = 2

km_1 = KMeans(n_clusters=n_components)

km_1.fit(X_sub_1)

y_predict_1 = km_1.predict(X_sub_1)
```

```
fig, ax = subplots(figsize=(4, 3))

ax.scatter(X_sub_1[:, 0], X_sub_1[:, 1],  c=y_predict_1, cmap='bwr_r');

show()
```



Demo of problems with Kmeans clustering due to specific data features:

https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_assumptions.html#sphx-glr-auto-examples-cluster-plot-kmeans-assumptions-py

Link to the paper highlighting pitfalls while clustering

[ ]:

2 Application: Ovarian Cancer Data

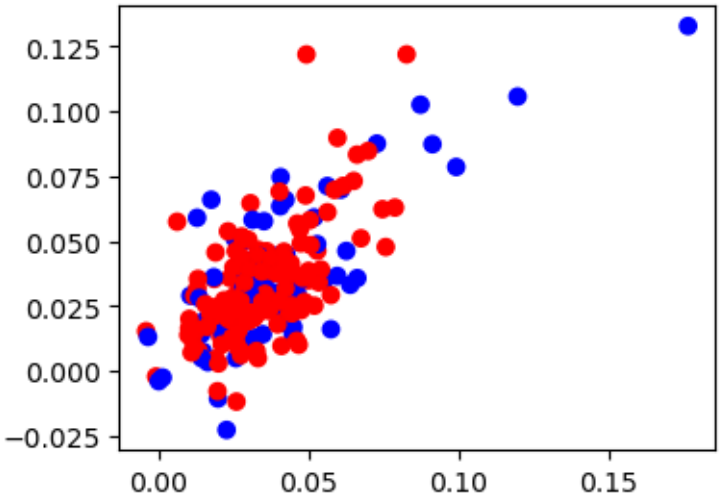
[22]:
```
df = read_csv('../Data/ovarian.txt', header=None)
labels = read_csv('../Data/ovarian_group.txt', header=None)

print(len(df), len(df.columns))

X = df.to_numpy()
y = (labels.to_numpy()).ravel()
```

```
216 4000
```

[23]:
```
n_components = 2
```

```
km_ov = KMeans(n_clusters=n_components)

km_ov.fit(X)

y_predict_ov = km_ov.predict(X)

scoring_ov = adjusted_rand_score(y, y_predict_ov)

print(scoring_ov)

fig, ax = subplots(figsize=(4, 3))

ax.scatter(X[:, 0], X[:, 1],  c=y_predict_ov, cmap='bwr_r');
```

```
0.19317411054208736
```



2.1 Combine with Supervised Learning

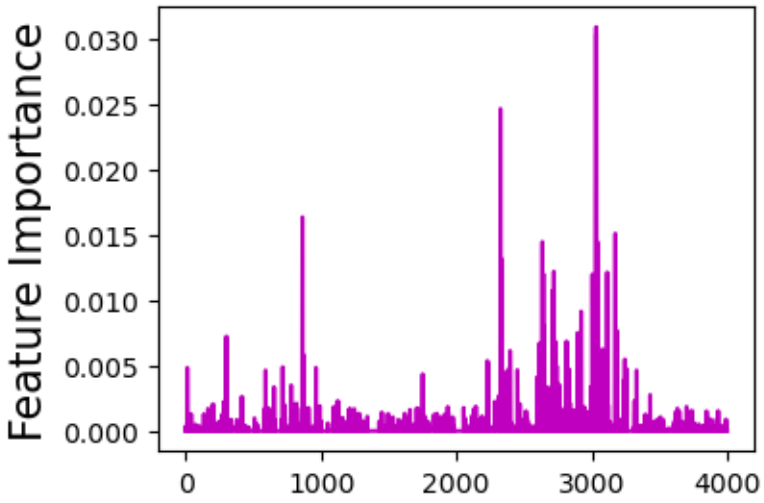
[24]:
```
from sklearn.model_selection import train_test_split

RANDOM_SEED = 1234

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.5,
                                                    random_state=RANDOM_SEED)
```

[25]:
```
from sklearn.ensemble import RandomForestClassifier

RANDOM_SEED = 999
```

```
clf = RandomForestClassifier(random_state=RANDOM_SEED)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

# Evaluating the score using test data:
score = clf.score(X_test, y_test)

print(round(score, 2))
```

```
0.91
```

[26]:
```
importances = clf.feature_importances_

fig, ax = subplots(figsize=(4, 3))

ax.plot(importances, color=('m'));
ax.set_ylabel('Feature Importance', fontsize=16);
```

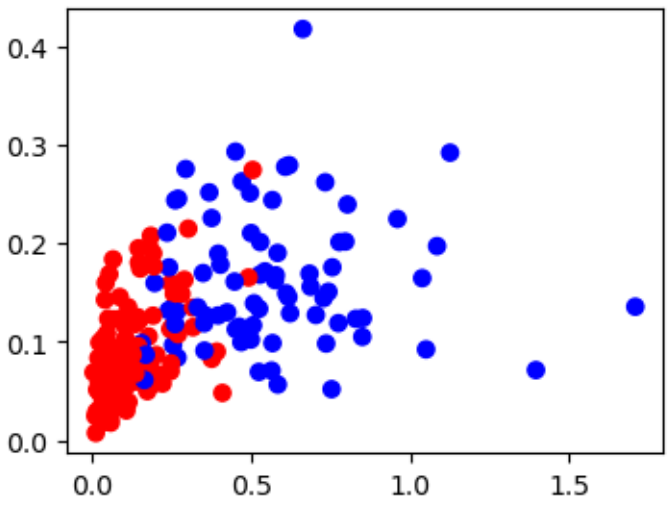


[27]:
```
importance_threshold = 0.01

imps_above = importances > importance_threshold

X_reduced = X[:, imps_above]

X_reduced.shape
```

[27]:
```
(216, 16)
```

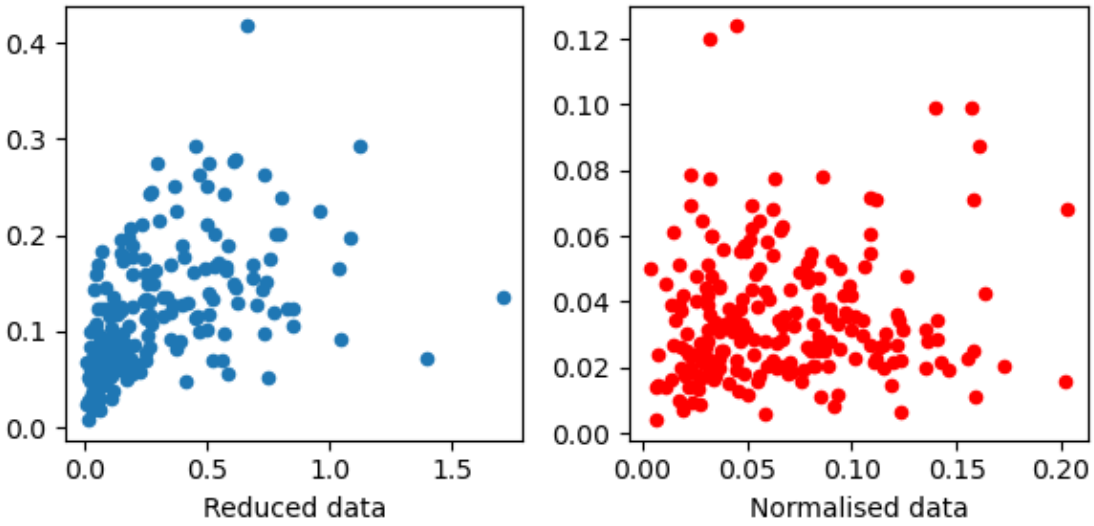
[28]:
```
n_components = 2

km = KMeans(n_clusters=n_components)

km.fit(X_reduced)

y_predict = km.predict(X_reduced)

from sklearn.metrics.cluster import adjusted_rand_score

scoring = adjusted_rand_score(y, y_predict)

print(scoring)

fig, ax = subplots(figsize=(4, 3))

ax.scatter(X_reduced[:, 0], X_reduced[:, 1],  c=y_predict, cmap='bwr_r');
```

```
0.3367962517486878
```



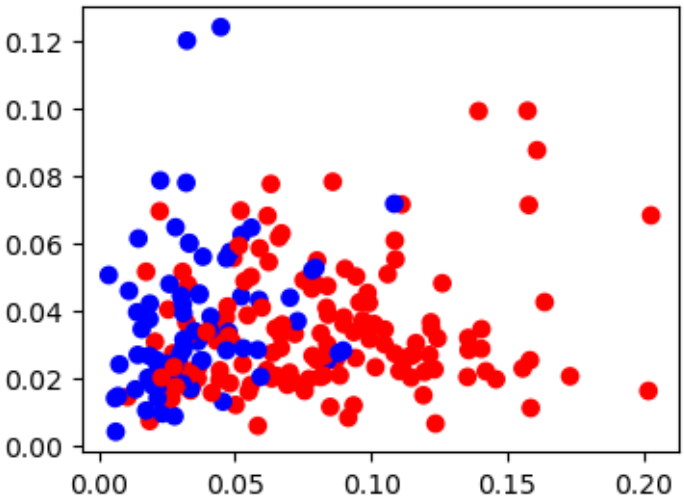
[34]:
```
from sklearn.preprocessing import Normalizer

norm_skl = Normalizer()

X_normed = norm_skl.fit_transform(X_reduced)

fig, ax = subplots(nrows=1, ncols=2, figsize=(6, 3))

ax[0].scatter(X_reduced[:, 0], X_reduced[:, 1], s=20)
ax[0].set_xlabel('Reduced data')
```

```
ax[1].scatter(X_normed[:, 0], X_normed[:, 1], s=20, c='r')
ax[1].set_xlabel('Normalised data');

fig.tight_layout()
```



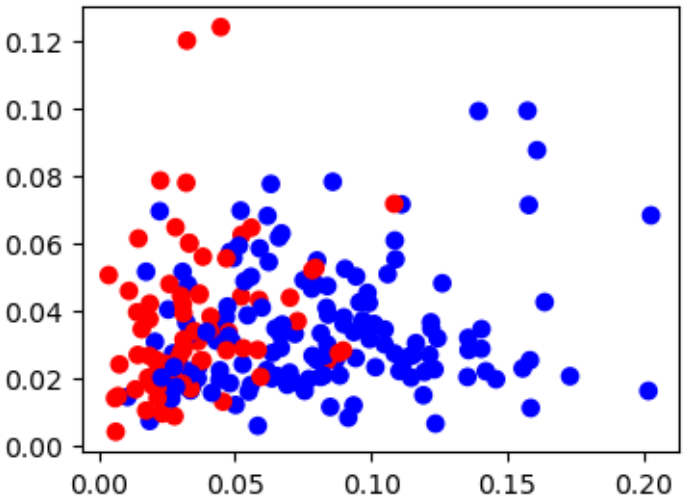
[35]:
```
km = KMeans(n_clusters=n_components)

km.fit(X_normed)

y_predict_km = km.predict(X_normed)

scoring = adjusted_rand_score(y, y_predict_km)

print(scoring)

fig, ax = subplots(figsize=(4, 3))

ax.scatter(X_normed[:, 0], X_normed[:, 1],  c=y_predict_km, cmap='bwr_r');
```

```
0.466540138034273
```



[36]:
```
from sklearn.mixture import GaussianMixture

gauss = GaussianMixture(n_components=n_components)

gauss.fit(X_normed)

y_predict_gauss = gauss.predict(X_normed)

scoring_gauss = adjusted_rand_score(y, y_predict_gauss)

print(scoring_gauss)

fig, ax = subplots(figsize=(4, 3))

ax.scatter(X_normed[:, 0], X_normed[:, 1],  c=y_predict_gauss, cmap='bwr');
```

```
0.4793554571602187
```



[37]:
```
adjusted_rand_score(y_predict_km, y_predict_gauss)
```

[37]: 0.9813234461288995

3 Conclusion

Clustering is an unsupervised way of dividing samples in a dataframe such that they fall into distinct categories. There are many ways of defining what constitutes a "distinct" category, normally some distance measure is used to define whether two points belong together or nor.

Unsupervised methods from scikit-learn are easy to use but results need critical quality testing. In addition to within-data assessments, this requires domain expertise, additional knowledge about the nature of the data.