Clustering by Kmeans and GMM

September 28, 2023

[159]:
```
import pandas as pd
import numpy as np
from sklearn import datasets

np.random.seed(254)

X, y = datasets.make_classification(1000, n_features=2, n_informative=2,␣
 ↪n_redundant=0, n_repeated=0, n_classes=3,
                                    n_clusters_per_class=1)

df= pd.DataFrame({"feature_1": X[:,0], "feature_2": X[:,1], "category":y},
                  columns=["feature_1", "feature_2", "category"])

df.shape
```

[159]:
```
(1000, 3)
```

[160]:
```
df.head()
```

[160]:
```
   feature_1  feature_2  category
0   1.316455   1.906756         0
1   0.499191  -1.668219         2
2   0.937946   0.834301         0
3   0.930180   0.648558         0
4   0.309252  -1.365109         2
```

[161]:
```
df.describe()
```

[161]:
```
         feature_1    feature_2     category
count  1000.000000  1000.000000  1000.000000
mean      0.332238    -0.299943     0.998000
std       1.142013     1.516528     0.818942
min      -2.234235    -5.113364     0.000000
25%      -0.779555    -1.399917     0.000000
50%       0.568081    -0.391351     1.000000
75%       1.221679     0.795395     2.000000
max       3.724172     4.034715     2.000000
```

[162]:

```
df.category.value_counts()
```

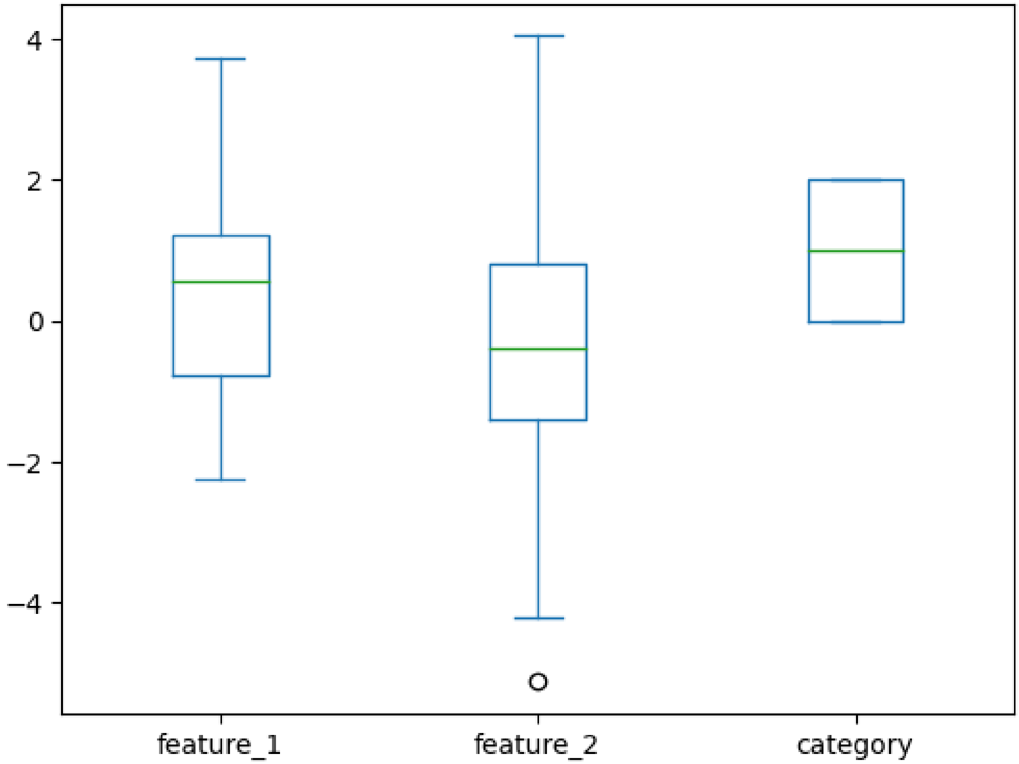
[162]:

```
0    336
2    334
1    330
Name: category, dtype: int64
```

[163]:

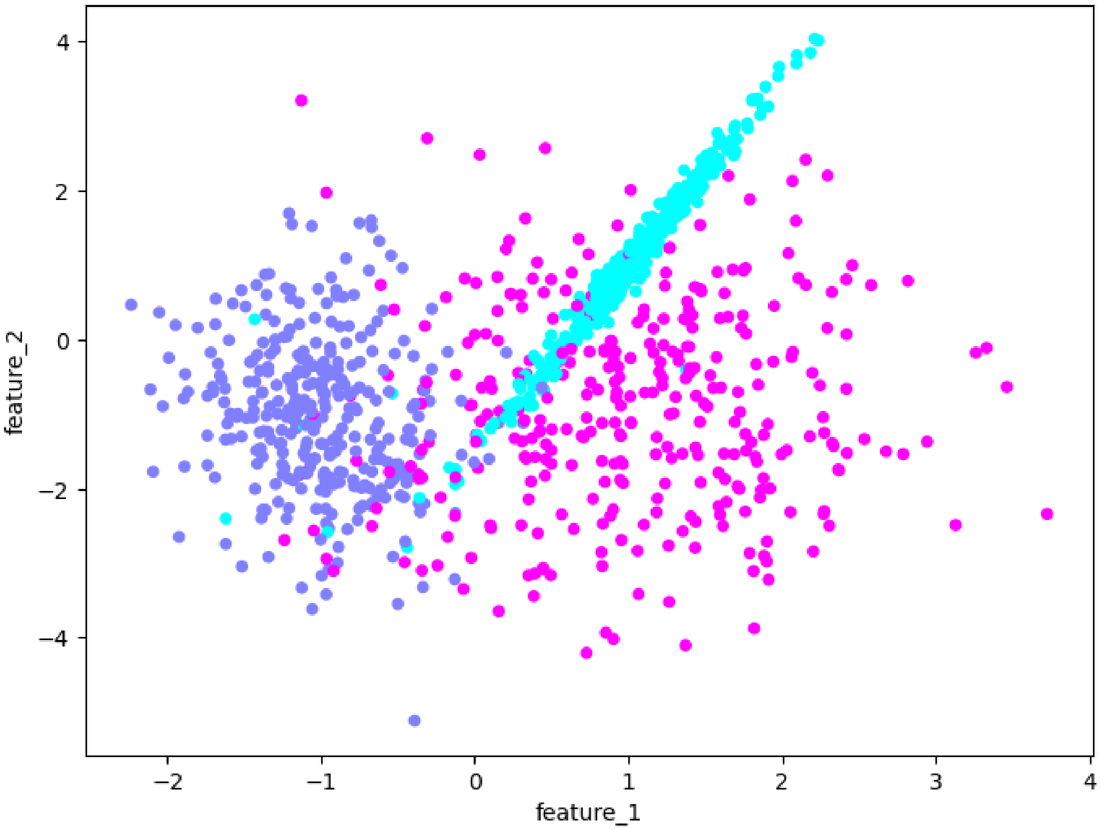
```
import matplotlib.pyplot as plt
import matplotlib as pylab

df.plot.box()
plt.show()
```



[164]:

```
df.plot.scatter("feature_1", "feature_2", c="category", cmap=pylab.cm.cool,␣
 ↪figsize=(8,6), colorbar=False)
plt.show()
```



[165]:

```
from sklearn import preprocessing
```

[166]:

```
scaler = preprocessing.StandardScaler()
```

[167]:

```
scaler.fit(df[["feature_1","feature_2"]])
```

[167]:

```
StandardScaler()
```

[168]:

```
standardized = scaler.transform(df[["feature_1","feature_2"]])
standardized
```

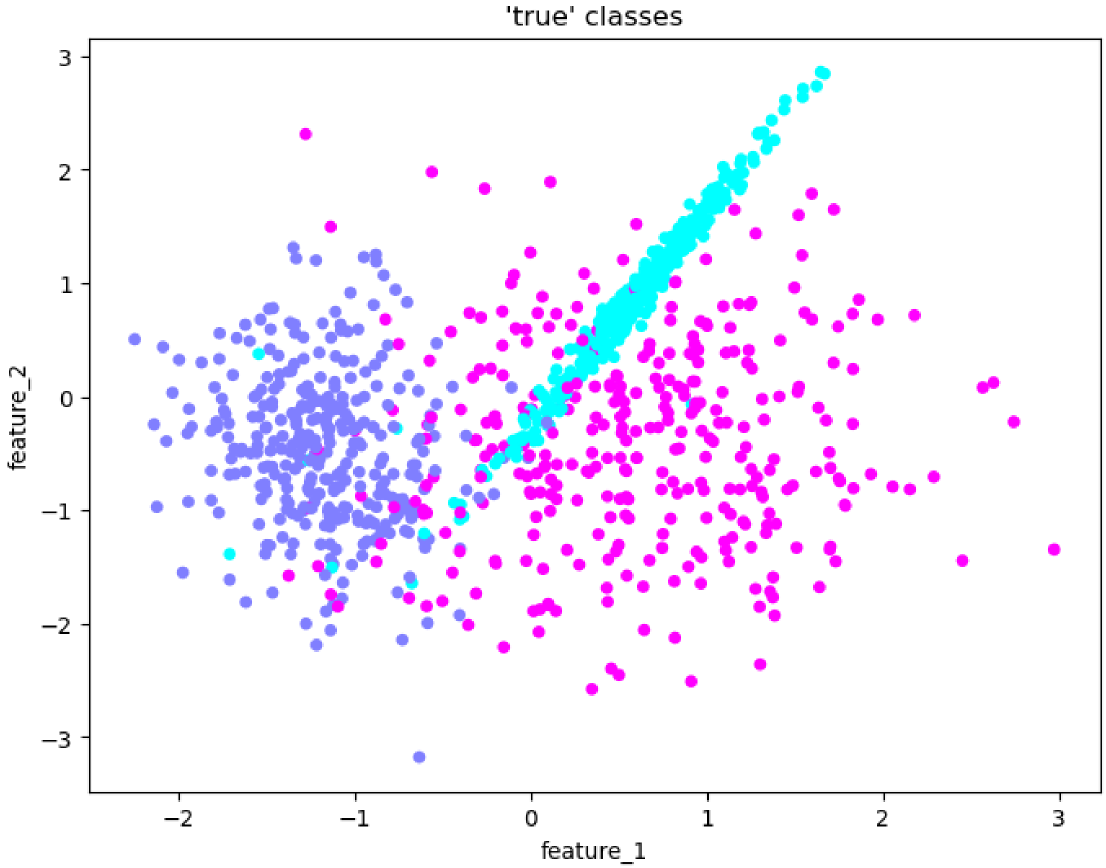
[168]:

```
array([[ 0.86225717,  1.45582779],
       [ 0.14626453, -0.90269403],
       [ 0.53065139,  0.74829594],
       …,
       [-1.11216242, -1.02386509],
       [ 1.82951322,  0.24260611],
       [ 0.12922246,  0.15564846]])
```

[169]:

```
df_scaled = df.copy()
df_scaled.loc[:, ["feature_1", "feature_2"]] = standardized
```

```
[170]: df_scaled.plot.scatter("feature_1", "feature_2", c="category", cmap=pylab.cm.
        ↪cool, title="'true' classes", figsize=(8,6), colorbar=False)
       plt.show()
```



1 K-mean clustering

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

```
[172]: kmeans = KMeans(n_clusters=3, random_state=42)
```

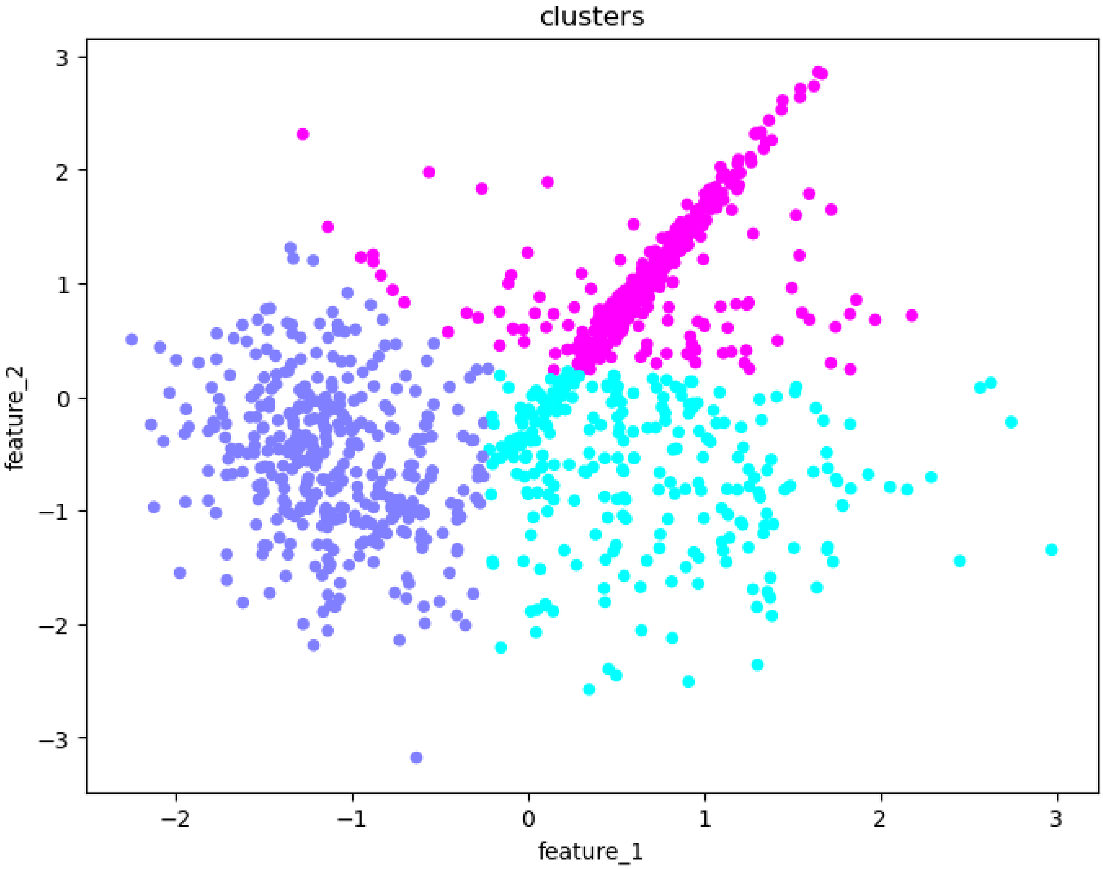
```
[173]: kmeans.fit(df_scaled[["feature_1","feature_2"]])
```

```
[173]: KMeans(n_clusters=3, random_state=42)
```

```
[174]: df_scaled["kmeans_cluster"] = kmeans.predict(df_scaled[["feature_1",␣
        ↪"feature_2"]])
```

[175]:

```
df_scaled.plot.scatter("feature_1", "feature_2", c="kmeans_cluster", cmap=pylab.cm.cool, figsize=(8,6), title="clusters", colorbar=False)
plt.show();
```



2 GMM Clustering

[176]:

```
from sklearn.mixture import GaussianMixture
from sklearn.decomposition import PCA
```

[177]:

```
gmm = GaussianMixture(n_components=3, random_state=42)
gmm.fit(df_scaled[["feature_1","feature_2"]])
```

[177]:

```
GaussianMixture(n_components=3, random_state=42)
```

[178]:

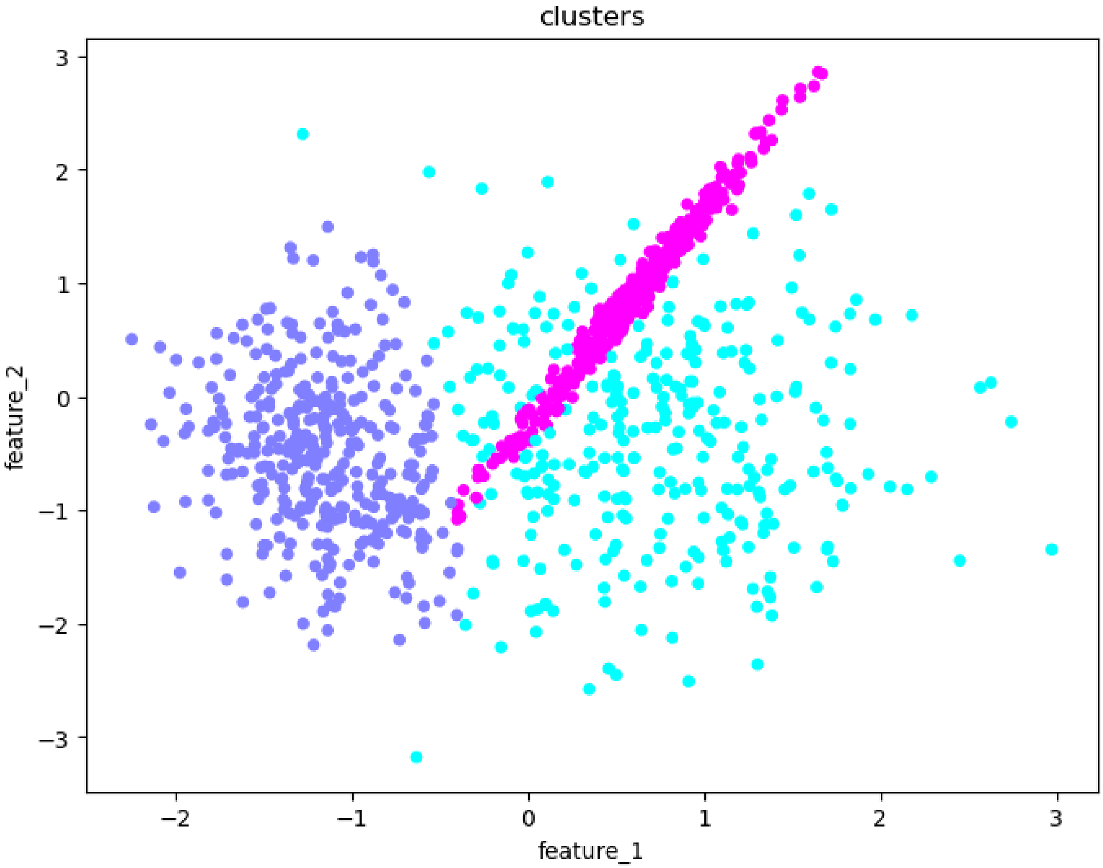
```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(df_scaled[["feature_1","feature_2"]])
```

[179]:

```
df['cluster'] = gmm.predict(df_scaled[["feature_1","feature_2"]])
```

[180]:

```
df_scaled.plot.scatter("feature_1", "feature_2", c=df["cluster"], cmap=pylab.cm.
 ↪cool, figsize=(8,6), title="clusters", colorbar=False)
plt.show();
```



[ ]:

```
```