ml-clustering

November 25, 2023

0.1 # Cluster Computing

0.2 Outline

Definition and Conceptual Overview

Types of Clustering

Centroid-Based

Distribution-Based

Hierarchical

Density-Based

Example Algorithms with Code (Using Python)

Centroid-Based

Distribution-Based

Hierarchical

Density-Based

Practice Exercises

0.3 Definition and Conceptual Overview

Clustering or Cluster analysis is a machine learning technique which groups unlabelled datasets.

It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them based on the presence and absence of similar patterns.

It is an Unsupervised Learning Method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.

After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML systems can use this id to simplify the processing of large and complex datasets.2

0.4 Types of Clustering

0.4.1 1. Centroid-Based Clustering

It is a type of clustering that divides the data into non-hierarchical groups. It is also known as Partitioning Clustering.

In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups. The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.1

```
<figure>
    <img src="https://media.geeksforgeeks.org/wp-content/cdn-uploads/20210722233746/KMeans-dens
    <figcaption align = "center"><i>Example Cluster Graph using K-Means where K=2.<sup>2</sup><
</figure>
```

0.4.2 2. Distribution-Based

In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution.1

This type of clustering algorithm is modeled using statistical distributions. It assumes that the data points in a cluster are generated from a particular probability distribution, and the algorithm aims to estimate the parameters of the distribution to group similar data points into clusters.2

```
<figure>
    <img src="https://media.geeksforgeeks.org/wp-content/cdn-uploads/20210722233721/EM-density-
    <figcaption align = "center"><i>Example Distribution-Based Clustering using Gaussian Distri
</figure>
```

0.4.3 3. Hierarchical Clustering

Hierarchical clustering, also known as Connectivity-based Clustering, is based on the principle that every object is connected to its neighbors depending on their proximity distance (degree of relationship).

The clusters are represented in extensive hierarchical structures separated by a maximum distance required to connect the cluster parts.

The clusters are represented as Dendrograms, where the X-axis represents the objects that do not merge while Y-axis is the distance at which clusters merge.

The similar data objects have minimal distance falling in the same cluster, and the dissimilar data objects are placed farther in the hierarchy. Mapped data objects correspond to a Cluster amid discrete qualities concerning the multidimensional scaling, quantitative relationships among data variables, or cross-tabulation in some aspects.

The hierarchical clustering may vary in the data flow chosen in the following categories.3

```
<figure>
    <img src="https://www.analytixlabs.co.in/blog/wp-content/uploads/2020/07/image-3-28-1-600x4
    <figcaption align = "center"><i>Example Hierarchical Cluster.<sup>3</sup></i></figcaption>
    <img src="https://www.analytixlabs.co.in/blog/wp-content/uploads/2020/07/image-4-17-1-600x3
    <figcaption align = "center"><i>Two types of Hierarchical Clustering.<sup>3</sup></i></figc
```

```
</figure>
```

3.1. Divisive Hierarchical Clustering3 This approach of hierarchical clustering follows a top-down approach where we consider that all the data points belong to one large cluster and try to divide the data into smaller groups based on a termination logic or a point beyond which there will be no further division of data points. This termination logic can be based on the minimum sum of squares of error inside a cluster, or for categorical data, the metric can be the GINI coefficient inside a cluster.

Hence, iteratively, we are splitting the data, which was once grouped as a single large cluster, into "n" number of smaller clusters to which the data points now belong.

It must be taken into account that this algorithm is highly "rigid" when splitting the clusters – meaning, once a clustering is done inside a loop, there is no way that the task can be undone.

3.2 Agglomerative Hierarchical Clustering3 Agglomerative is quite the contrary to Divisive, where all the "N" data points are considered to be a single member of "N" clusters that the data is comprised into. We iteratively combine these numerous "N" clusters to a fewer number of clusters, let's say "k" clusters, and hence assign the data points to each of these clusters accordingly.

This approach is a bottom-up one, and also uses a termination logic in combining the clusters.

This logic can be a number-based criterion (no more clusters beyond this point) or a distance criterion (clusters should not be too far apart to be merged) or a variance criterion (increase in the variance of the cluster being merged should not exceed a threshold, Ward Method)

0.4.4 4. Density-Based Clustering

The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected.

This algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters. The dense areas in data space are divided from each other by sparser areas.

These algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.1

```
<figure>
    <img src="https://static.javatpoint.com/tutorial/machine-learning/images/clustering-in-mach
    <figcaption align = "center"><i>Example Density-Based Clustering.<sup>1</sup></i></figcapti
</figure>
```

</div

The clusters formed vary in arbitrary shapes and sizes and contain a maximum degree of homogeneity due to similar density. This clustering approach includes the noise and outliers in the datasets effectively.

When performing most of the clustering, we take two major assumptions: the data is devoid of any noise and the shape of the cluster so formed is purely geometrical (circular or elliptical).

The fact is, data always has some extent of inconsistency (noise) which cannot be ignored. Added to that, we must not limit ourselves to a fixed attribute shape. It is desirable to have arbitrary

shapes to not to ignore any data points. These are the areas where density-based algorithms have proven their worth.3

0.5 Example Clustering Algorithms (Using Python)

0.5.1 Test Data (California Housing Data from KAGGLE)

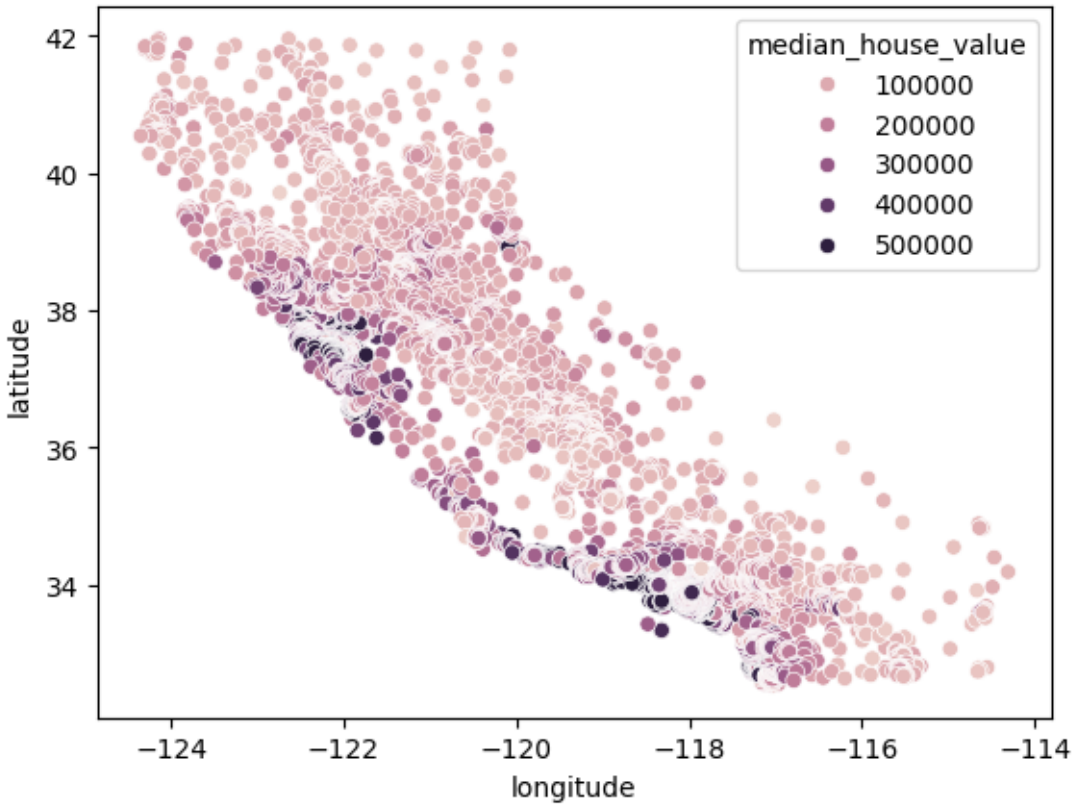
[2]:
```
import pandas as pd

home_data = pd.read_csv('housing.csv', usecols = ['longitude', 'latitude',␣
 ↪'median_house_value'])
home_data.tail(10)
```

[2]:

	longitude	latitude	median_house_value
20630	-121.32	39.29	112000.0
20631	-121.40	39.33	107200.0
20632	-121.45	39.26	115600.0
20633	-121.53	39.19	98300.0
20634	-121.56	39.27	116800.0
20635	-121.09	39.48	78100.0
20636	-121.21	39.49	77100.0
20637	-121.22	39.43	92300.0
20638	-121.32	39.43	84700.0
20639	-121.24	39.37	89400.0

[3]:
```
import seaborn as sns

sns.scatterplot(data = home_data, x = 'longitude', y = 'latitude', hue =␣
 ↪'median_house_value')
```

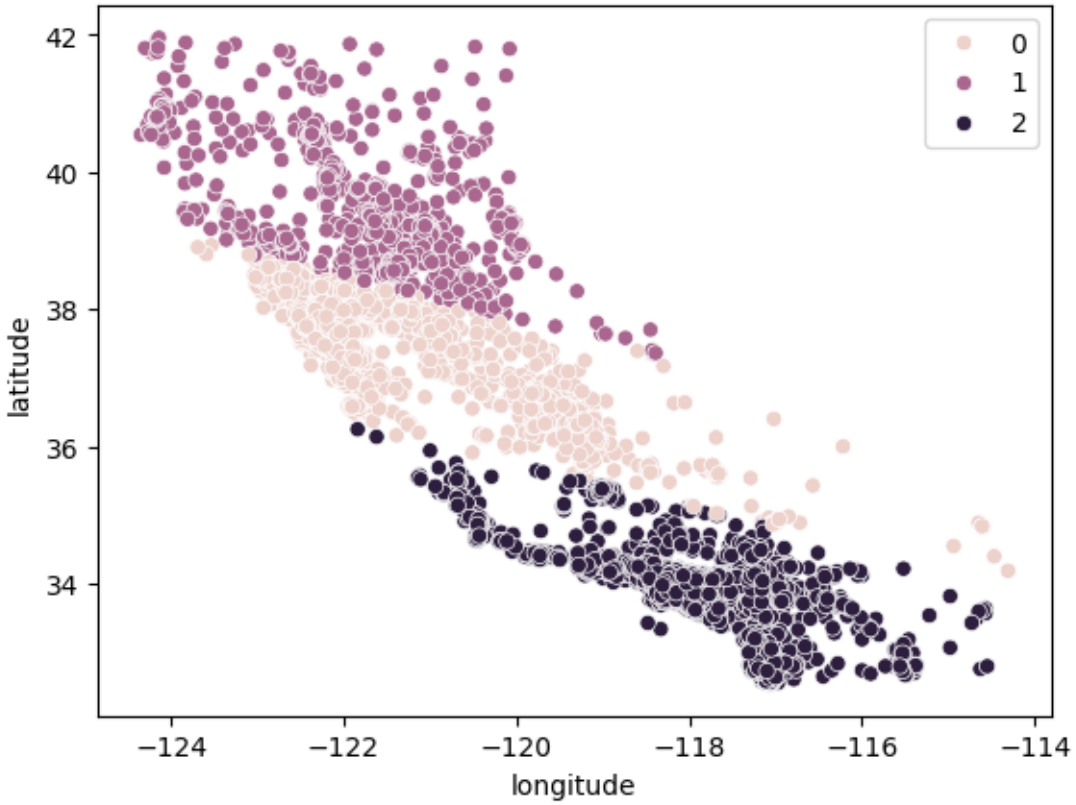
[3]:
```
<Axes: xlabel='longitude', ylabel='latitude'>
```



0.5.2 1. Centroid-Based: K-Means Clustering 4

[10]:
```
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.cluster import KMeans #Note: sklearn import KMeans command might␣
 ↪not be supported on later versions of Python

X_train, X_test, y_train, y_test = train_test_split(home_data[['latitude',␣
 ↪'longitude']], home_data[['median_house_value']], test_size=0.33,␣
 ↪random_state=0)

X_train_norm = preprocessing.normalize(X_train)
X_test_norm = preprocessing.normalize(X_test)

kmeans = KMeans(n_clusters = 3, random_state = 0, n_init='auto')
kmeans.fit(X_train_norm)

sns.scatterplot(data = X_train, x = 'longitude', y = 'latitude', hue = kmeans.
 ↪labels_)
```

[10]: <Axes: xlabel='longitude', ylabel='latitude'>

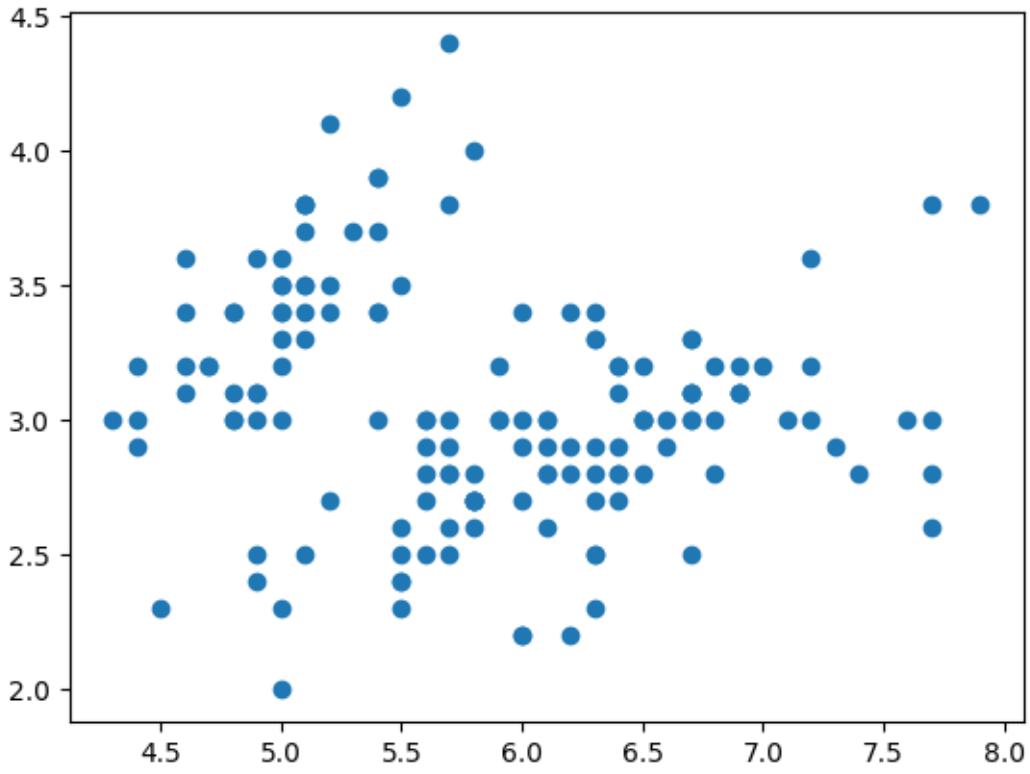


0.5.3 2. Distribution-Based: Gaussian Mixture Model (GMM)

Example Data set

[10]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas import DataFrame
from sklearn import datasets
from sklearn.mixture import GaussianMixture

# load the iris dataset
iris = datasets.load_iris()

# select first two columns
X = iris.data[:, :2]

# turn it into a dataframe
d = pd.DataFrame(X)
```

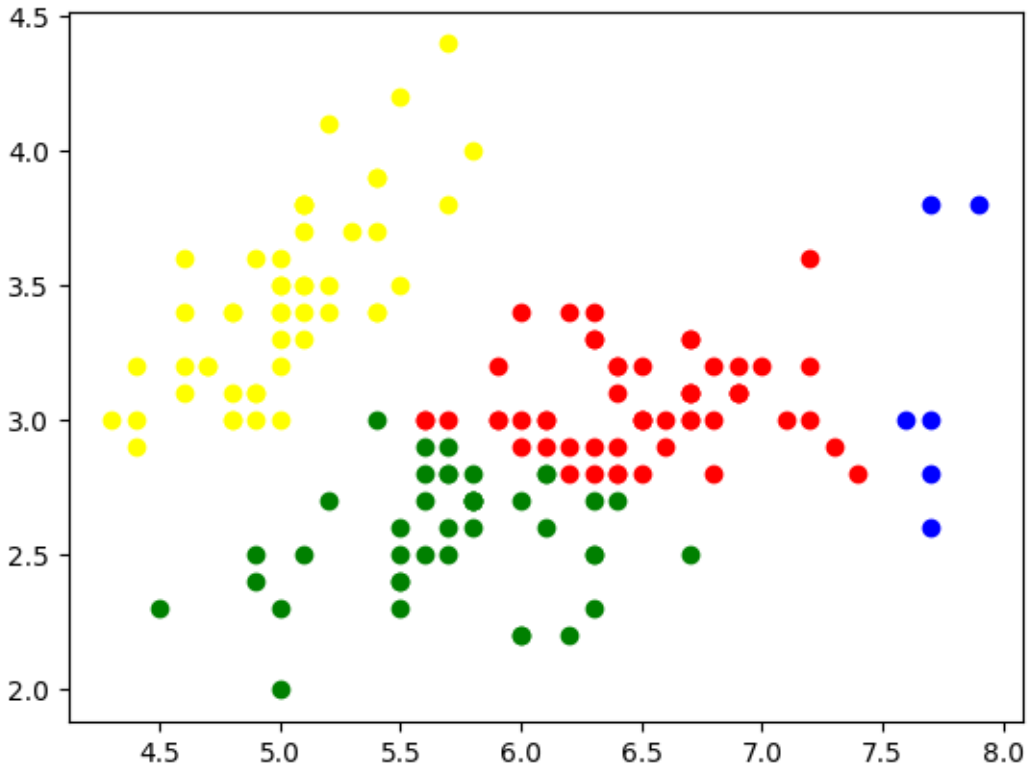
```
# plot the data
plt.scatter(d[0], d[1])
plt.show()
```



Gaussian Mixture Model

[11]:
```
gmm = GaussianMixture(n_components = 4)

# Fit the GMM model for the dataset
# which expresses the dataset as a
# mixture of 3 Gaussian Distribution
gmm.fit(d)

# Assign a label to each sample
labels = gmm.predict(d)
d['labels']= labels
d0 = d[d['labels']== 0]
d1 = d[d['labels']== 1]
d2 = d[d['labels']== 2]
d3 = d[d['labels']== 3]

# plot three clusters in same plot
```

```
plt.scatter(d0[0], d0[1], c ='r')
plt.scatter(d1[0], d1[1], c ='yellow')
plt.scatter(d2[0], d2[1], c ='g')
plt.scatter(d3[0], d3[1], c ='blue')

plt.show()
```



0.5.4 3. Hierarchical Clustering: Agglomerative (Bottom-Up) and Divisive (Top-Down)

Test Data (Iris Plant Dataset from UC IRVINE5)

[14]:
```
from ucimlrepo import fetch_ucirepo

# fetch dataset
iris = fetch_ucirepo(id=53)

# data (as pandas dataframes)
X = iris.data.features
y = iris.data.targets

# metadata
```

```
#print(iris.metadata)
print(X)
# variable information
#print(iris.variables)
```

```
     sepal length  sepal width  petal length  petal width
0             5.1          3.5           1.4          0.2
1             4.9          3.0           1.4          0.2
2             4.7          3.2           1.3          0.2
3             4.6          3.1           1.5          0.2
4             5.0          3.6           1.4          0.2
..            …            …             …            …
145           6.7          3.0           5.2          2.3
146           6.3          2.5           5.0          1.9
147           6.5          3.0           5.2          2.0
148           6.2          3.4           5.4          2.3
149           5.9          3.0           5.1          1.8

[150 rows x 4 columns]
```

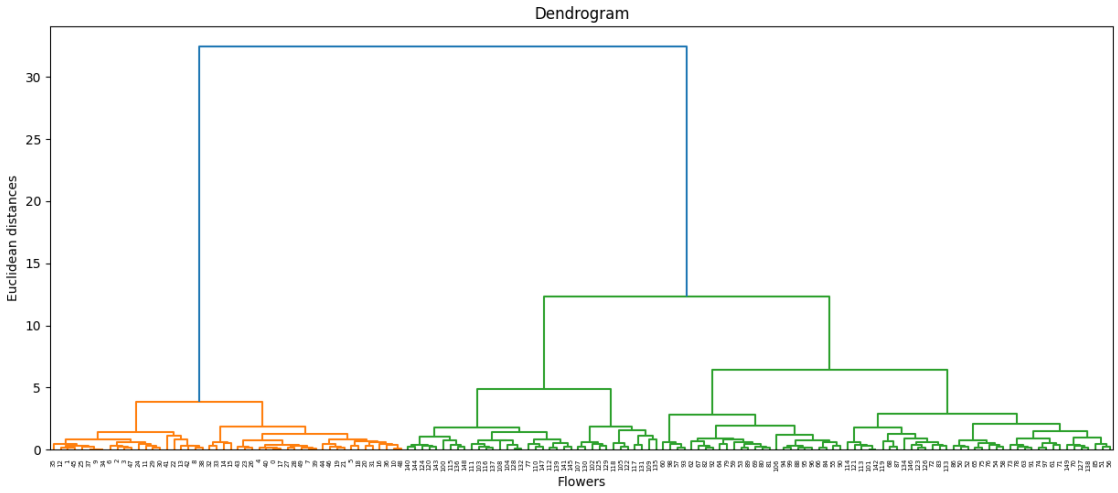
Hierarchical Clustering

[19]:
```
import pandas as pd
import numpy as np

#import the class
from sklearn.cluster import AgglomerativeClustering

#data_path = "/content/Iris.csv"
#read the data
#df = pd.read_csv(data_path)
#select only feature columns
#X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
print(X)

#instantiate the model
model = AgglomerativeClustering(n_clusters = 3, metric = 'euclidean', linkage =␣
 ↪'ward') #changed affinity to metric FutureWarning: Attribute `affinity` was␣
 ↪deprecated in version 1.2 and will be removed in 1.4. Use `metric` instead
#fit the model and predict the clusters
y_pred = model.fit_predict(X)
```

```
     sepal length  sepal width  petal length  petal width
0             5.1          3.5           1.4          0.2
1             4.9          3.0           1.4          0.2
2             4.7          3.2           1.3          0.2
3             4.6          3.1           1.5          0.2
4             5.0          3.6           1.4          0.2
```

```
..     …    …    …    …
145  6.7  3.0  5.2  2.3
146  6.3  2.5  5.0  1.9
147  6.5  3.0  5.2  2.0
148  6.2  3.4  5.4  2.3
149  5.9  3.0  5.1  1.8

[150 rows x 4 columns]
```

[20]:
```
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt

plt.figure(figsize=(15,6))
plt.title('Dendrogram')
plt.xlabel('Flowers')
plt.ylabel('Euclidean distances')
#create linkage matrix
link_matrix = linkage(X,method='ward')
dendrogram = dendrogram(link_matrix)
plt.savefig("iris.png")
plt.show()
```

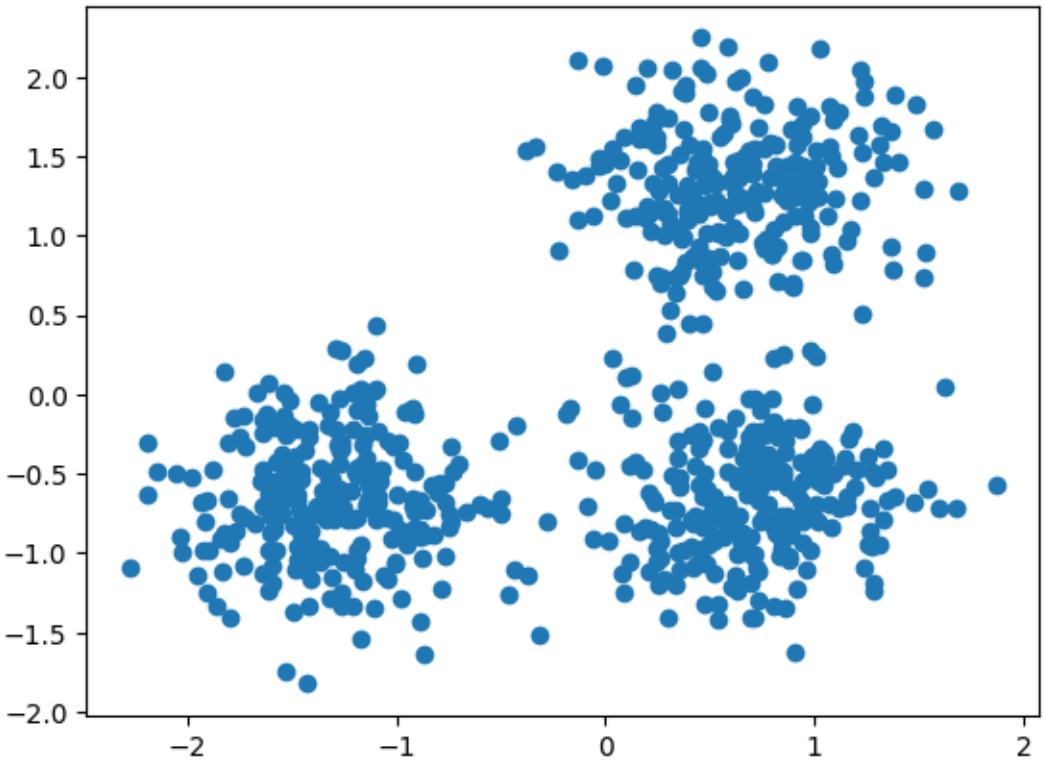


0.5.5 4. Density-Based Clustering

Test Dataset6

[23]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets import make_blobs
```

```
from sklearn.preprocessing import StandardScaler
from sklearn import datasets


centers = [[1, 1], [-1, -1], [1, -1]]
X, labels_true = make_blobs(
    n_samples=750, centers=centers, cluster_std=0.4, random_state=0
)

X = StandardScaler().fit_transform(X)
plt.scatter(X[:, 0], X[:, 1])
plt.show()
```

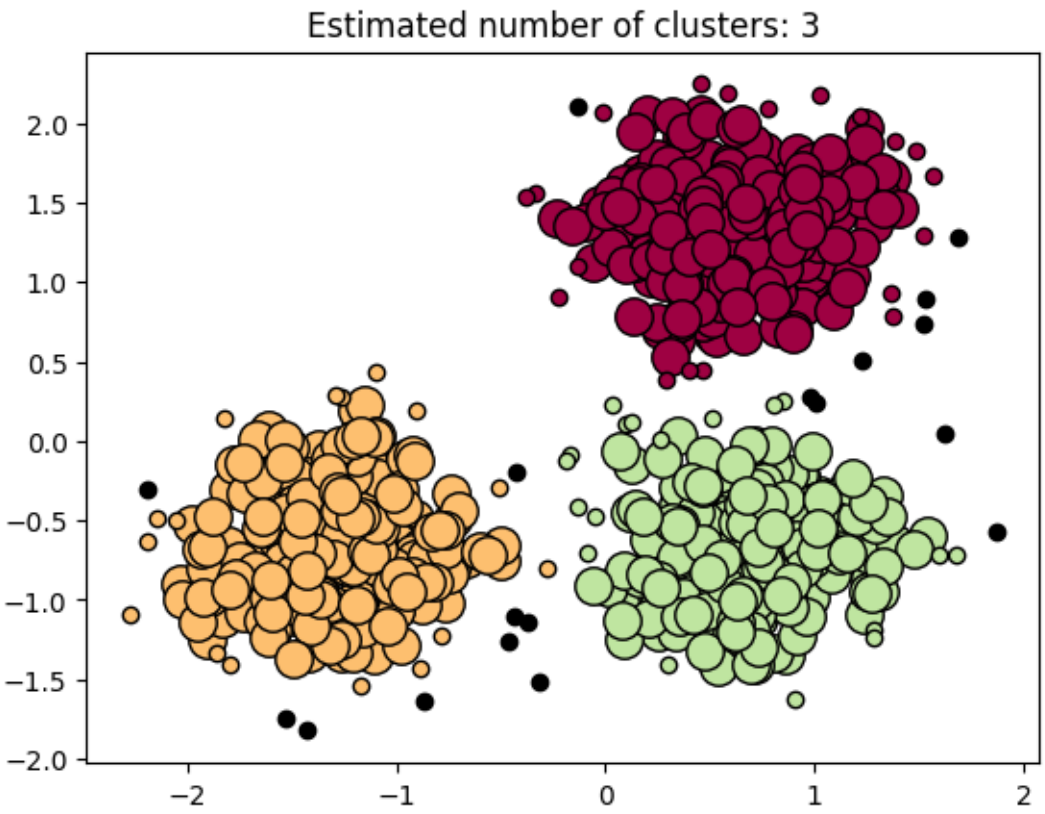


[26]:
```
import numpy as np

from sklearn import metrics
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=0.3, min_samples=10).fit(X)
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
```

```
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print("Estimated number of clusters: %d" % n_clusters_)
print("Estimated number of noise points: %d" % n_noise_)
```

```
Estimated number of clusters: 3
Estimated number of noise points: 18
```

[25]:
```
unique_labels = set(labels)
core_samples_mask = np.zeros_like(labels, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True

colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1,␣
 ↪len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]

    class_member_mask = labels == k

    xy = X[class_member_mask & core_samples_mask]
    plt.plot(
        xy[:, 0],
        xy[:, 1],
        "o",
        markerfacecolor=tuple(col),
        markeredgecolor="k",
        markersize=14,
    )

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(
        xy[:, 0],
        xy[:, 1],
        "o",
        markerfacecolor=tuple(col),
        markeredgecolor="k",
        markersize=6,
    )

plt.title(f"Estimated number of clusters: {n_clusters_}")
plt.show()
```



0.6 References:

Java T Point

Geeks for Geeks

ANALYTIX LABS

Datacamp

UC Irvine Machine Learning Repository

scikit-learn Website

Created by: Lyberius Ennio F. Taruc For: CPE647 Class Date: 2023-11-25 Version: 1.0

[ ]: