Experiment 14

Dhruv Singhal || 500075346 || R177219074 || AIML || Sem5

DBSCAN Implementation

Importing Required Libraries ¶

In [1]:
```
from sklearn.cluster import DBSCAN
import pandas as pd
import matplotlib.pyplot as plt
import warnings
from sklearn.preprocessing import StandardScaler
warnings.filterwarnings('ignore')
```

Data

for this experiment we have use make_blobs datsets which are already in sklearn.datasets

sklearn.datasets.make_blobs(n_samples=100, n_features=2, *, centers=None, cluster_std=1.0, center_box=(- 10.0, 10.0), shuffle=True, random_state=None, return_centers=False) Generate isotropic Gaussian blobs for clustering.

In [2]:
```
from sklearn.datasets import make_blobs
```

In [3]:
```
X,y=make_blobs(n_samples=1000, n_features=2, centers=[[0.5, 2], [-1, -1], [1.5, -1]],
               cluster_std=0.5, center_box=(- 10.0, 10.0), shuffle=True, random_state=42)
```

In [4]:
```
X.shape
```

Out[4]:
```
(1000, 2)
```

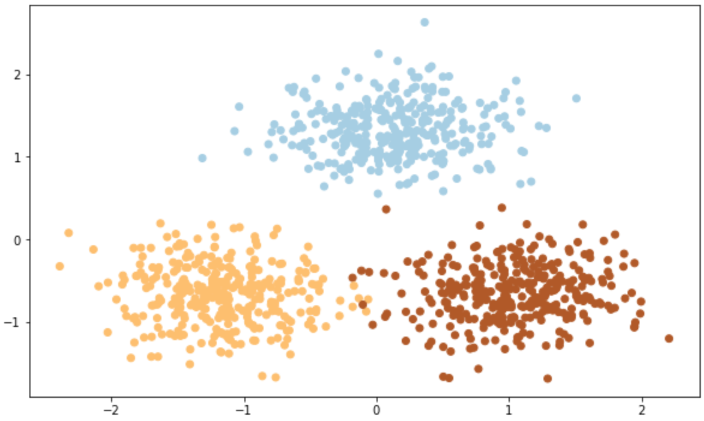
In [5]:
```
y.shape
```

Out[5]:
```
(1000,)
```

Data Preprocessing and Visualization

In [6]:
```
X = StandardScaler().fit_transform(X)
plt.figure(figsize=(10,6))
plt.scatter(X[:,0], X[:,1], c=y, cmap='Paired')
```

Out[6]:
```
<matplotlib.collections.PathCollection at 0x22707191e88>
```



Model Building

DBSCAN(Density based spatial clustering of application with noise) is a clustering based algorithm.It is an unsupervised algorithm that separates the data point into specific groups which have similar density.

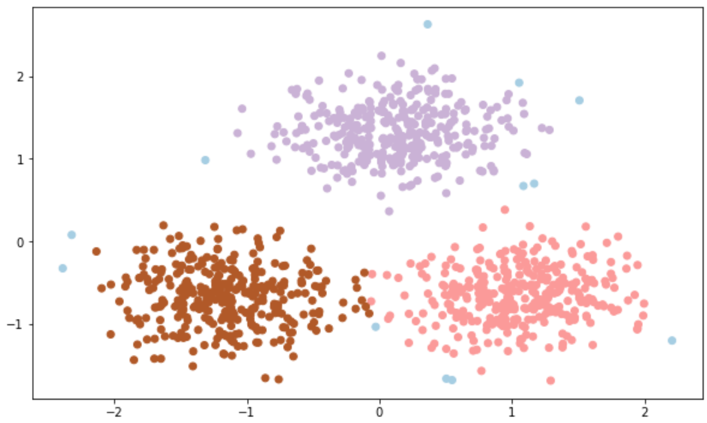
- Some common parameters of SVM:
  - Epsilon(eps): It refers to the maximum distance b/w the samples for one to be considered as in the neighbourhood of other.
  - min_samples: It refer to the minimum number of point in the neighbourhood of the point to be considered as core point.
- Terminology:
  - Core point: A point is a core point if there are at least minPts number of points (including the point itself) in its surrounding area with radius eps.
  - Boundry point: A point is a boundry point if it is reachable from a core point and there are less than min_samples number of points within its surrounding area.
  - Noise point: Point that is neither core nor boundry.

In [7]:

```
db = DBSCAN(eps=0.45, min_samples=50)
db.fit(X)
y_pred = db.fit_predict(X)
plt.figure(figsize=(10,6))
plt.scatter(X[:,0], X[:,1],c=y_pred, cmap='Paired')
```

Out[7]:

```
<matplotlib.collections.PathCollection at 0x227099de8c8>
```



In [8]:

```
#%% Applying DBSCAN model
clustering = DBSCAN(eps=3, min_samples=10)
clusters=clustering.fit_predict(X, y, sample_weight=None)
core=clustering.core_sample_indices_
components=clustering.components_
print("min samples ",10)
print("number of core points:",len(core))
print("number of components :",len(components))
```

```
min samples  10
number of core points: 1000
number of components : 1000
```

In [9]:

```
#%%  Applying DBSCAN model with different min_samples value
clustering = DBSCAN(eps=3, min_samples=15)
clusters=clustering.fit_predict(X, y, sample_weight=None)
core=clustering.core_sample_indices_
components=clustering.components_
print("min samples ",15)
print("number of core points:",len(core))
print("number of components :",len(components))
```

```
min samples  15
number of core points: 1000
number of components : 1000
```

In [10]:

```
#%%  Applying DBSCAN model with different min_samples value
clustering = DBSCAN(eps=1.56, min_samples=20)
clusters=clustering.fit_predict(X, y, sample_weight=None)
core=clustering.core_sample_indices_
components=clustering.components_
print("min samples ",20)
print("number of core points:",len(core))
print("number of components :",len(components))
```

```
min samples  20
number of core points: 1000
number of components : 1000
```

In [ ]:

In [ ]: