Submitted By : 211042 Noman masood Khan

Section : A

Submitted TO : Sir Mahaz

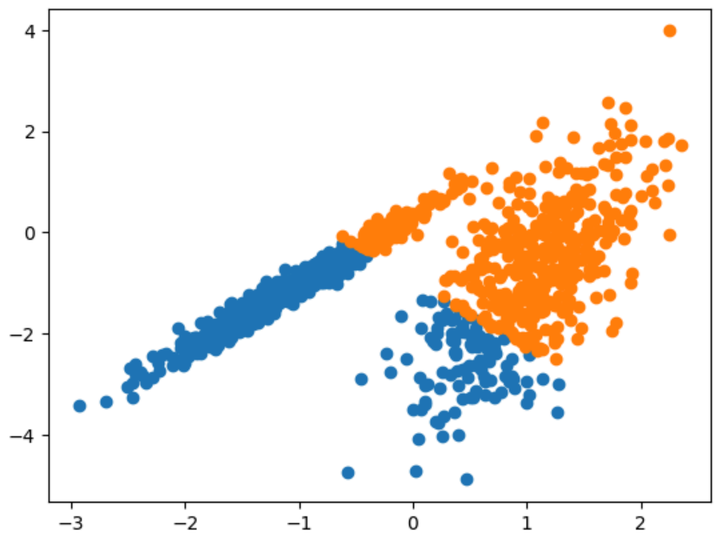
LAB Number : Lab 9

Air University Islamabad

1. K-means clustering algorithm

```
from numpy import unique, where
from matplotlib import pyplot
from sklearn.datasets import make_classification
from sklearn.cluster import KMeans

# initialize the data set we'll work with
training_data, _ = make_classification(
    n_samples=1000,
    n_features=2,
    n_informative=2,
    n_redundant=0,
    n_clusters_per_class=1,
    random_state=4
)

# define the model
kmeans_model = KMeans(n_clusters=2)

# train the model
kmeans_model.fit(training_data)

# assign each data point to a cluster
kmeans_result = kmeans_model.predict(training_data)

# get all of the unique clusters
kmeans_clusters = unique(kmeans_result)

# plot the K-means clusters
for kmeans_cluster in kmeans_clusters:
    index = where(kmeans_result == kmeans_cluster)
    pyplot.scatter(training_data[index, 0], training_data[index, 1])

# show the K-means plot
pyplot.show()
```

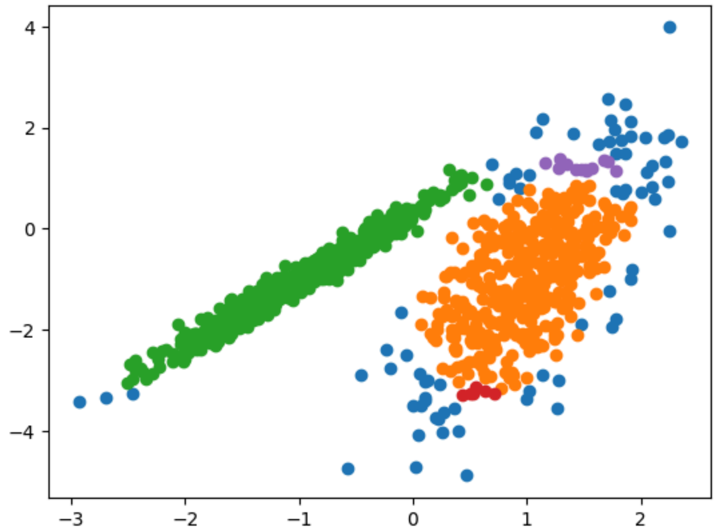
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: T
  warnings.warn(
```



DBSCAN clustering algorithm

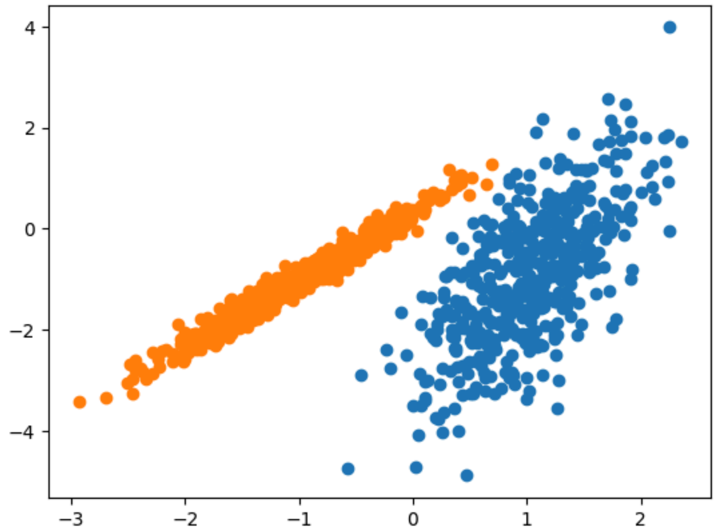
Description: DBSCAN is a density-based clustering algorithm suitable for finding outliers and handling oddly shaped data.

```
from numpy import unique, where
from matplotlib import pyplot
from sklearn.datasets import make_classification
from sklearn.cluster import DBSCAN

# initialize the data set we'll work with
training_data, _ = make_classification(
    n_samples=1000,
    n_features=2,
    n_informative=2,
    n_redundant=0,
    n_clusters_per_class=1,
    random_state=4
)

# define the model
dbscan_model = DBSCAN(eps=0.25, min_samples=9)

# train the model
dbscan_model.fit(training_data)

# assign each data point to a cluster
dbscan_result = dbscan_model.labels_

# get all of the unique clusters
dbscan_clusters = unique(dbscan_result)

# plot the DBSCAN clusters
for dbscan_cluster in dbscan_clusters:
    index = where(dbscan_result == dbscan_cluster)
    pyplot.scatter(training_data[index, 0], training_data[index, 1])

# show the DBSCAN plot
pyplot.show()
```



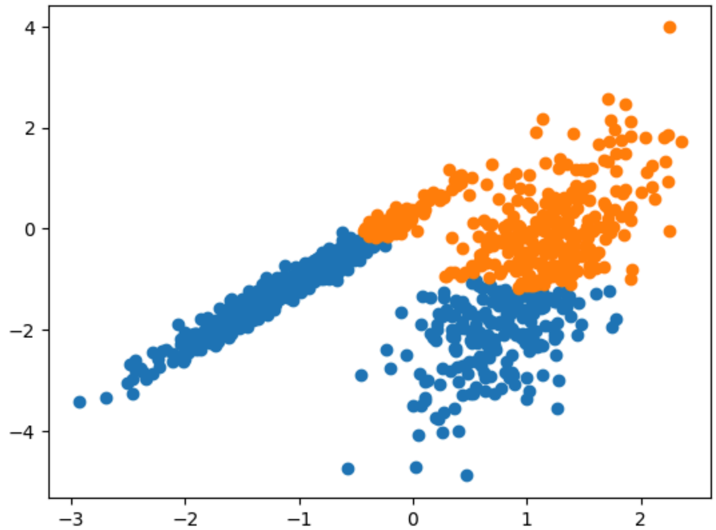
3. Gaussian Mixture Model algorithm

```
from numpy import unique, where
from matplotlib import pyplot
from sklearn.datasets import make_classification
from sklearn.mixture import GaussianMixture

# initialize the data set we'll work with
training_data, _ = make_classification(
    n_samples=1000,
    n_features=2,
    n_informative=2,
    n_redundant=0,
    n_clusters_per_class=1,
    random_state=4
)

# define the model
gaussian_model = GaussianMixture(n_components=2)

# train the model
gaussian_model.fit(training_data)

# assign each data point to a cluster
gaussian_result = gaussian_model.predict(training_data)

# get all of the unique clusters
gaussian_clusters = unique(gaussian_result)

# plot Gaussian Mixture the clusters
for gaussian_cluster in gaussian_clusters:
    index = where(gaussian_result == gaussian_cluster)
    pyplot.scatter(training_data[index, 0], training_data[index, 1])

# show the Gaussian Mixture plot
pyplot.show()
```



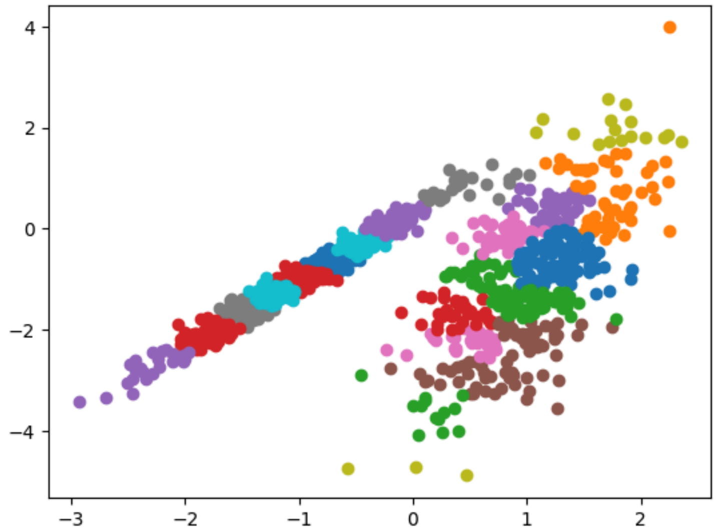
BIRCH algorithm

```
from numpy import unique, where
from matplotlib import pyplot
from sklearn.datasets import make_classification
from sklearn.cluster import Birch

# initialize the data set we'll work with
training_data, _ = make_classification(
    n_samples=1000,
    n_features=2,
    n_informative=2,
    n_redundant=0,
    n_clusters_per_class=1,
    random_state=4
)

# define the model
birch_model = Birch(threshold=0.03, n_clusters=2)

# train the model
birch_model.fit(training_data)

# assign each data point to a cluster
birch_result = birch_model.predict(training_data)

# get all of the unique clusters
birch_clusters = unique(birch_result)

# plot the BIRCH clusters
for cluster in birch_clusters:
    # get data points that fall in this cluster
    index = where(birch_result == cluster)
    # make the plot
    pyplot.scatter(training_data[index, 0], training_data[index, 1])

# show the BIRCH plot
pyplot.show()
```



6. Affinity Propagation clustering algorithm

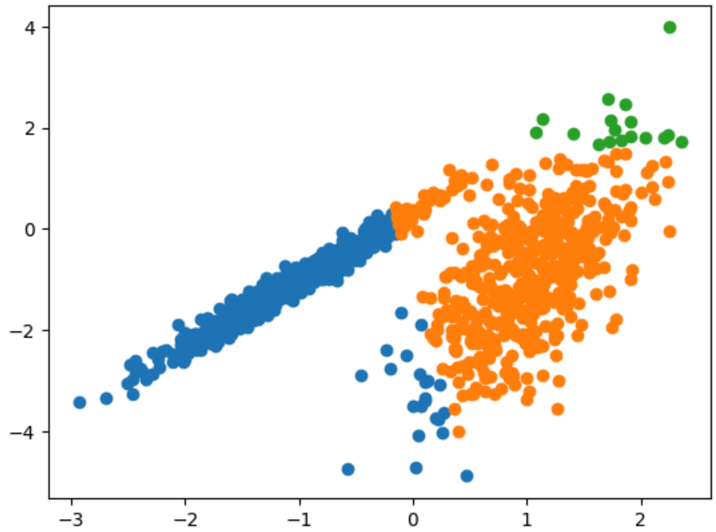
```
from numpy import unique, where
from matplotlib import pyplot
from sklearn.datasets import make_classification
from sklearn.cluster import AffinityPropagation

# initialize the data set we'll work with
training_data, _ = make_classification(
    n_samples=1000,
    n_features=2,
    n_informative=2,
    n_redundant=0,
    n_clusters_per_class=1,
    random_state=4
)

# define the model
model = AffinityPropagation(damping=0.7)

# train the model
model.fit(training_data)

# assign each data point to a cluster
result = model.predict(training_data)

# get all of the unique clusters
clusters = unique(result)

# plot the clusters
for cluster in clusters:
    # get data points that fall in this cluster
    index = where(result == cluster)
    # make the plot
    pyplot.scatter(training_data[index, 0], training_data[index, 1])

# show the plot
pyplot.show()
```



Double-click (or enter) to edit

7 Mean-Shift clustering algorithm

```
from numpy import unique, where
from matplotlib import pyplot
from sklearn.datasets import make_classification
from sklearn.cluster import MeanShift

# initialize the data set we'll work with
training_data, _ = make_classification(
    n_samples=1000,
    n_features=2,
    n_informative=2,
    n_redundant=0,
    n_clusters_per_class=1,
    random_state=4
)

# define the model
mean_model = MeanShift()

# train the model and assign each data point to a cluster
mean_result = mean_model.fit_predict(training_data)

# get all of the unique clusters
mean_clusters = unique(mean_result)

# plot Mean-Shift clusters
for cluster in mean_clusters:
    # get data points that fall in this cluster
    index = where(mean_result == cluster)
    # make the plot
    pyplot.scatter(training_data[index, 0], training_data[index, 1])

# show the Mean-Shift plot
pyplot.show()
```



8. OPTICS algorithm

```
from numpy import unique, where
from matplotlib import pyplot
from sklearn.datasets import make_classification
from sklearn.cluster import OPTICS

# initialize the data set we'll work with
training_data, _ = make_classification(
    n_samples=1000,
    n_features=2,
    n_informative=2,
    n_redundant=0,
    n_clusters_per_class=1,
    random_state=4
)

# define the model
optics_model = OPTICS(eps=0.75, min_samples=10)

# train the model and assign each data point to a cluster
optics_result = optics_model.fit_predict(training_data)

# get all of the unique clusters
optics_clusters = unique(optics_result)

# plot OPTICS clusters
for cluster in optics_clusters:
    # get data points that fall in this cluster
    index = where(optics_result == cluster)
    # make the plot
    pyplot.scatter(training_data[index, 0], training_data[index, 1])

# show the OPTICS plot
pyplot.show()
```

Agglomerative Hierarchical clustering algorithm