CLUSTERING

What is clustering

Clustering is a type of unsupervised learning technique where the objective is to arrive at conclusions based on the patterns found within unlabeled input data. This technique is mainly used to segregate large data into subgroups in order to make informed decisions.

Clustering types

- **Centroid-based models**: These models are based on algorithms that define a centroid for each cluster, which is updated constantly by an iterative process.The data points are assigned to the cluster where their proximity to the centroid is minimized. Ex: k-means algorithm

- **Density-based models**: As the name suggests, these models define clusters by their density in the data space. This means that areas with a high density of data points will become clusters, which are typically separated from one another by low-density areas. Ex: DBSCAN algorithm.

Clustering types

- **Connectivity-based models**:
  This model's approach to similarity is based on proximity in a data space.
  The creation of clusters can be done by assigning all data points to a single cluster and then partitioning the data into smaller clusters as the distance between data points increases.
  Likewise, the algorithm can also start by assigning each data point an individual cluster, and then aggregating data points that are close by. Ex: hierarchical clustering.

- **Distribution-based models**: Models that fall into this category are based on the probability that all the data points from a cluster follow the same distribution, such as a Gaussian distribution. Ex: Gaussian Mixture algorithm.

Applications of clustering

- Search engine results
- Recommendation programs
- Image recognition
- Market segmentation

K-means

The k-means algorithm works through an iterative process that involves the following steps:

1. Based on the number of clusters defined by the user, the centroids are generated either by setting initial estimates or by randomly choosing them from the data points. This step is known as initialization.
2. All the data points are assigned to the nearest cluster in the data space by measuring their respective distances from the centroid, known as the assignment step. The objective is to minimize the squared Euclidean distance, which can be defined by the following formula:

   $$\min \text{dist}(c,x)^2$$

   Here, c represents a centroid, x refers to a data point, and dist() is the Euclidean distance.

K-means

3. Centroids are calculated again by computing the mean of all the data points belonging to a cluster. This step is known as the update step.

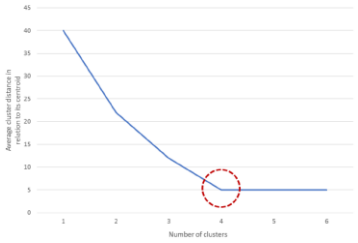
   Steps 2 and 3 are repeated in an iterative process until a criterion is met. This criterion can be as follows:

   - The number of iterations defined.
   - The data points do not change from cluster to cluster.
   - The Euclidean distance is minimized.

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Choosing k for k-means

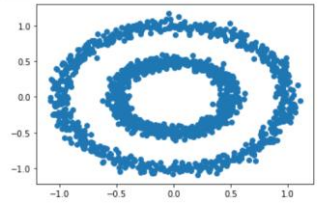
Plot the average distance between the data points and the cluster centroid against the number of clusters. The appropriate number of clusters corresponds to the breaking point of the plot, where the rate of decrease drastically changes. In the following diagram, the dotted circle represents the ideal number of clusters:



k-means

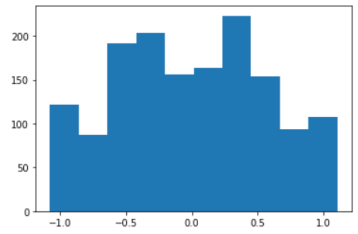
```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

In [2]: data = pd.read_csv("circles.csv")
        plt.scatter(data.iloc[:,0], data.iloc[:,1])
        plt.show()
```



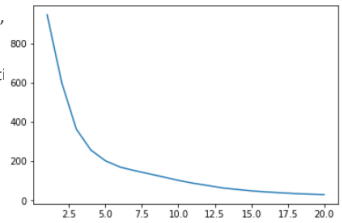
k-means

```
In [3]: plt.hist(data.iloc[:,0])
        plt.show()
```



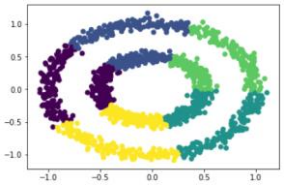
k-means

```
from sklearn.cluster import KMeans
ideal_k = []
for i in range(1,21):
  est_kmeans = KMeans(n_clusters=i,
  est_kmeans.fit(data)
  ideal_k.append([i,est_kmeans.inerti
ideal_k = np.array(ideal_k)
plt.plot(ideal_k[:,0],ideal_k[:,1])
plt.show()
```



k-means

```
est_kmeans = KMeans(n_clusters=5,
random_state=0)
est_kmeans.fit(data)
pred_kmeans = est_kmeans.predict(data)
plt.scatter(data.iloc[:,0], data.iloc[:,1],
c=pred_kmeans)
plt.show()
```



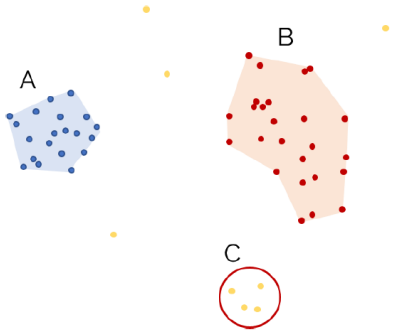
DBSCAN

- The density-based spatial clustering of applications with noise (DBSCAN) algorithm groups together points that are close to each other (with many neighbors) and marks those points that are further away with no close neighbors as outliers.
- The DBSCAN algorithm requires two main parameters: epsilon and the minimum number of observations.

  Epsilon, also known as eps, is the maximum distance that defines the radius within which the algorithm searches for neighbors.

  The **minimum number of observations**, on the other hand, refers to the number of data points required to form a high-density area (min_samples)

DBSCAN



DBSCAN

- According to this, each data point can be classified as follows:

  A core point: A point that has at least the minimum number of data points within its eps radius.

  A border point: A point that is within the eps radius of a core point, but does not have the required number of data points within its own radius.

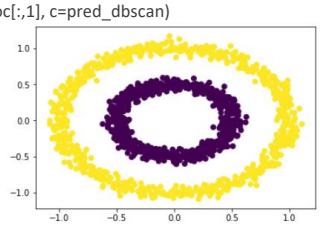
  A noise point: All points that do not meet the preceding descriptions.

DBSCAN

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

DBSCAN

```
from sklearn.cluster import DBSCAN
est_dbscan = DBSCAN(eps=0.1)
pred_dbscan = est_dbscan.fit_predict(data)
plt.scatter(data.iloc[:,0], data.iloc[:,1], c=pred_dbscan)
plt.show()
```



Evaluating the Performance of Clusters

The **Silhouette Coefficient Score** calculates the mean distance between each point and all the other points of a cluster (a), as well as the mean distance between each point and all the other points of its nearest clusters (b). It relates both of them according to the following equation:

$$s = (b - a) / \max(a,b)$$

The **Calinski–Harabasz Index** was created to measure the relationship between the variance of each cluster and the variance of all clusters. More specifically, the variance of each cluster is the mean square error of each point with respect to the centroid of that cluster. On the other hand, the variance of all clusters refers to the overall inter-cluster variance.

Evaluating the Performance of Clusters

The **Silhouette Coefficient Score** calculates the mean distance between each point and all the other points of a cluster (a), as well as the mean distance between each point and all the other points of its nearest clusters (b). It relates both of them according to the following equation:

$$s = (b - a) / \max(a,b)$$

The **Calinski–Harabasz Index** was created to measure the relationship between the variance of each cluster and the variance of all clusters. More specifically, the variance of each cluster is the mean square error of each point with respect to the centroid of that cluster. On the other hand, the variance of all clusters refers to the overall inter-cluster variance.

Evaluating the Performance of Clusters

```
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabaz_score
kmeans_score = silhouette_score(data, pred_kmeans,metric='euclidean')
dbscan_score = silhouette_score(data, pred_dbscan,metric='euclidean')
print(kmeans_score, dbscan_score)
```

Evaluating the Performance of Clusters

```
kmeans_score = calinski_harabasz_score(data, pred_kmeans)
dbscan_score = calinski_harabasz_score(data, pred_dbscan)
print(kmeans_score, dbscan_score)
```