Clustering - introduction

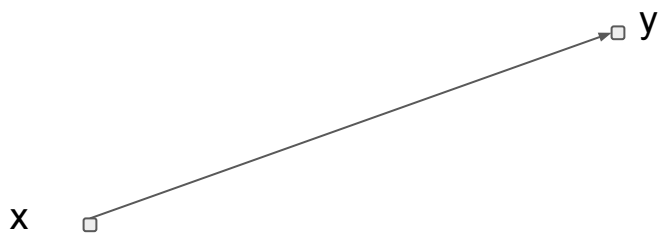
Marcin Wierzbiński

Euclidean distance

The *distance* (more precisely the *Euclidean distance*) between two points of a Euclidean space is the length of the translation vector that maps one point to the other; that is

$$d(x,y) = ||x-y||$$

where x,y are two points.

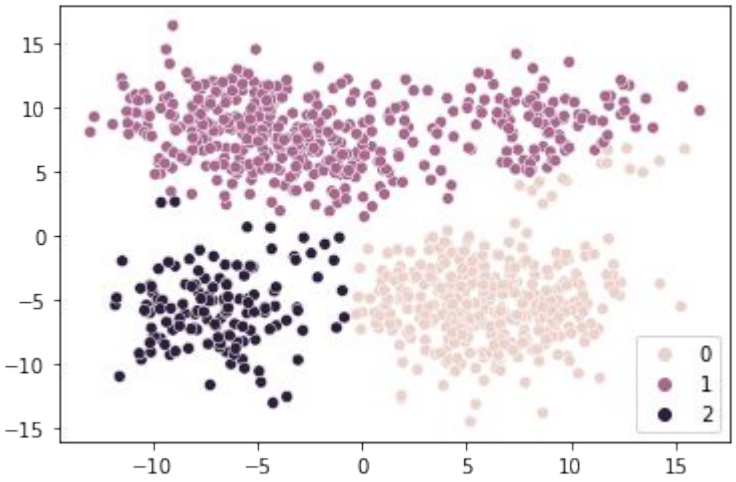


Euclidean distance numpy

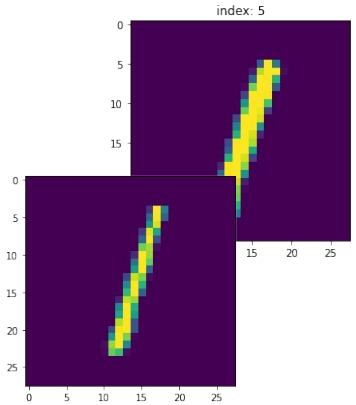
```
import numpy as np

point1 = np.array([1, 2, 3])
point2 = np.array([1, 1, 1])

# subtracting vector
temp = point1 - point2
# doing dot product for finding sum of the squares
sum_sq = np.dot(temp, temp)
# Doing squareroot and printing Euclidean distance
print(np.sqrt(sum_sq))
```

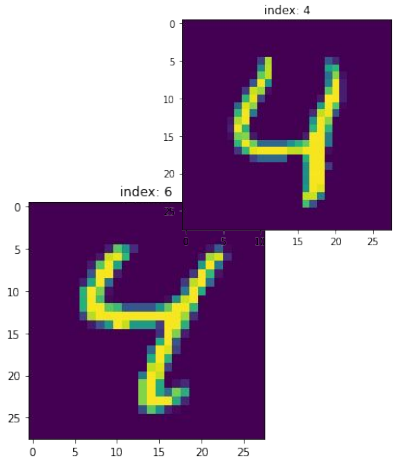
Clustering

Is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense) to each other than to those in other groups (clusters).



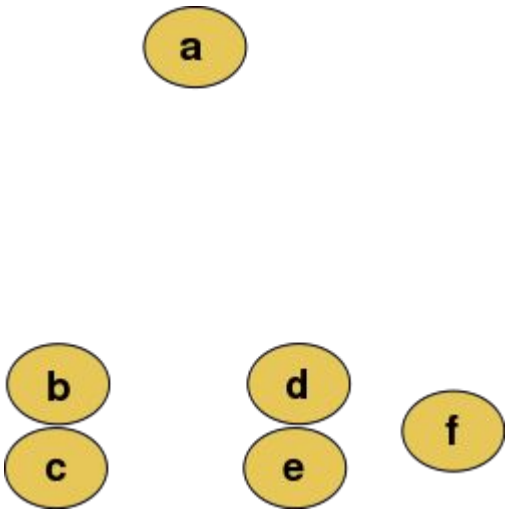
Example 1



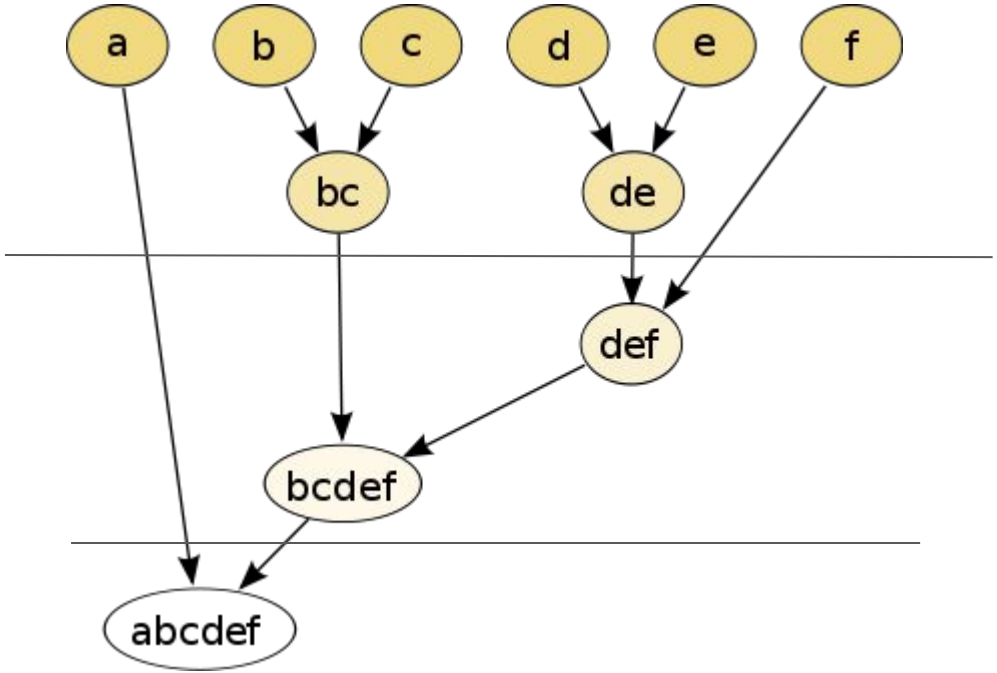


Example 2

Agglomerative clustering example



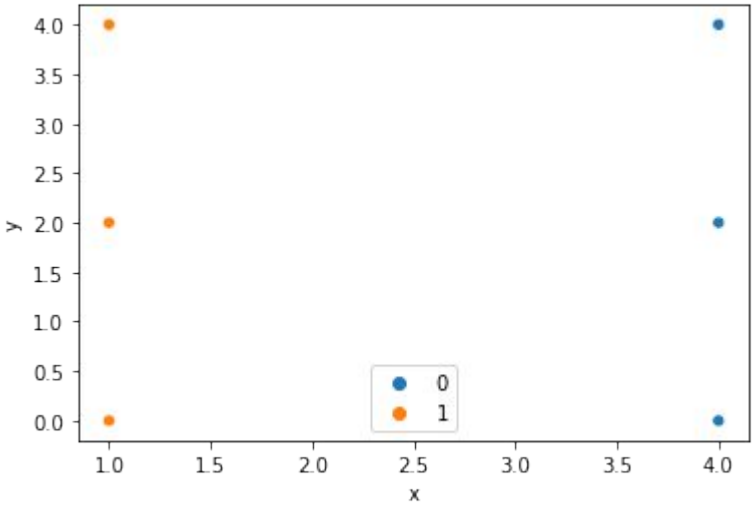
For example, suppose this data is to be clustered, and the Euclidean distance is the distance metric.



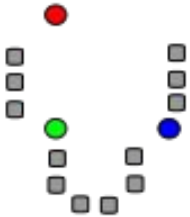
The hierarchical clustering dendrogram.

Basic example

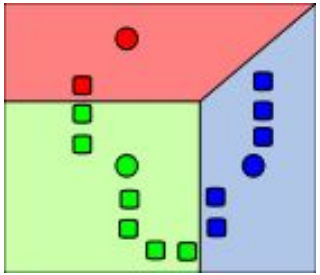
```
from sklearn.cluster import AgglomerativeClustering
import numpy as np
X = np.array([[1, 2], [1, 4], [1, 0],
              [4, 2], [4, 4], [4, 0]])
clustering = AgglomerativeClustering().fit(X)
clustering
clustering.labels_
clustering.n_clusters
```



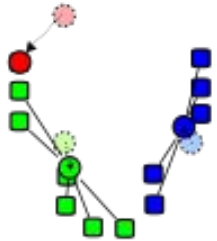
K means algorithm



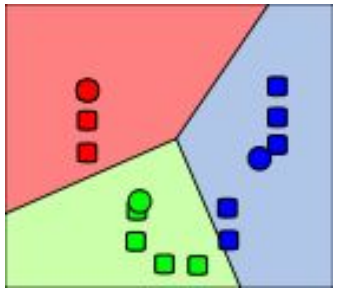
1. k initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).



2. *k* clusters are created by associating every observation with the nearest mean.



3. The centroid of each of the *k* clusters becomes the new mean.

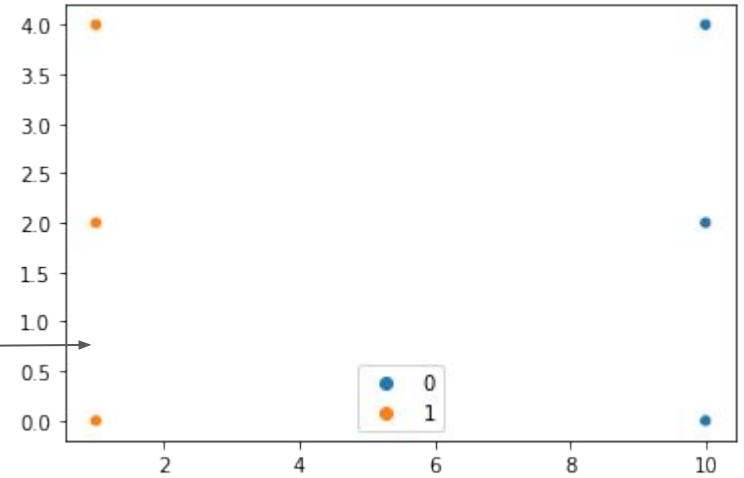


4. Steps 2 and 3 are repeated until convergence has been reached.

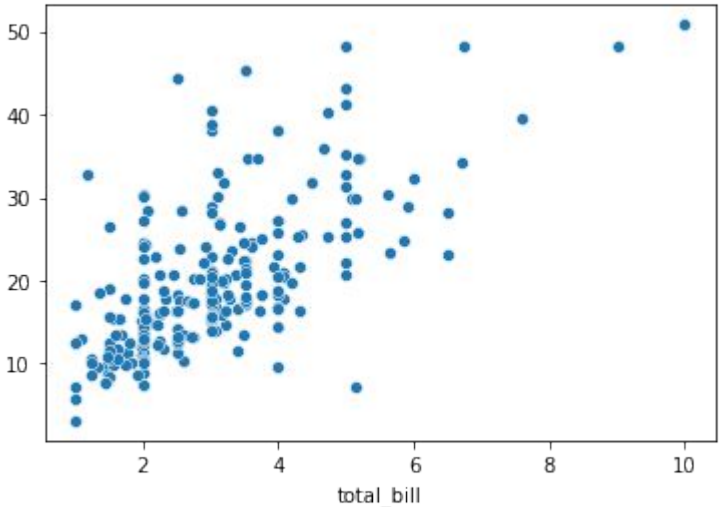
Basic example

```
from sklearn.cluster import KMeans
import numpy as np
X = np.array([[1, 2], [1, 4], [1, 0],
              [10, 2], [10, 4], [10, 0]])
kmeans = KMeans(n_clusters=2, random_state=0, n_init="auto").fit(X)
kmeans.labels_
kmeans.predict([[0, 0], [12, 3]]) #array([1, 0], dtype=int32)
kmeans.cluster_centers_
```

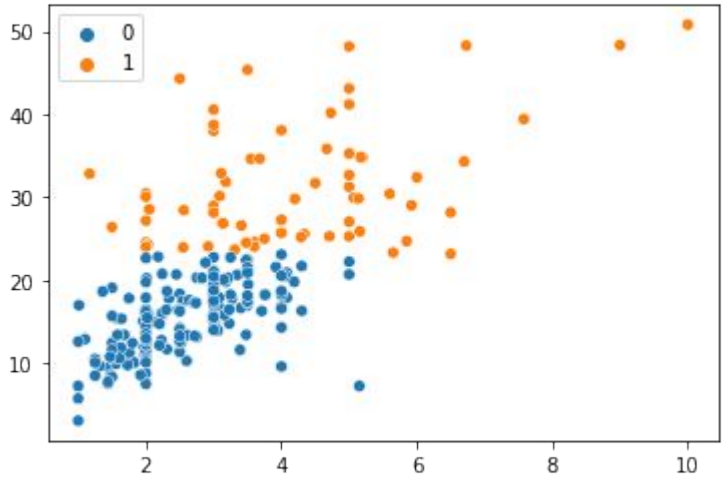
```
import seaborn as sns
sns.scatterplot(X[:,0],X[:,1], hue=kmeans.labels_)
```



Tips clustering K-means

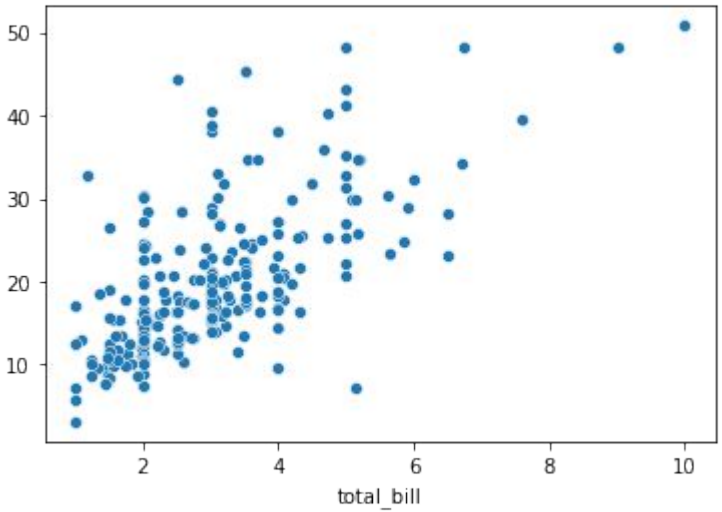


original data

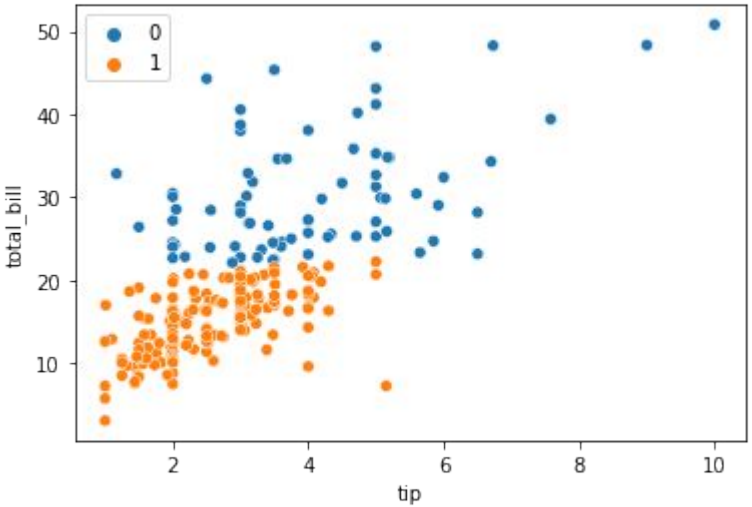



```
kmeans = KMeans(n_clusters=2, random_state=0,
n_init="auto").fit(X)
```

Tips clustering Agglomerative

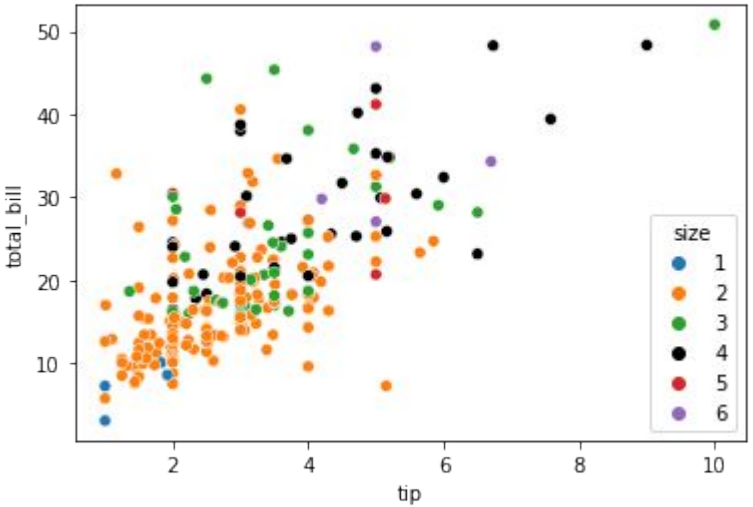


original data

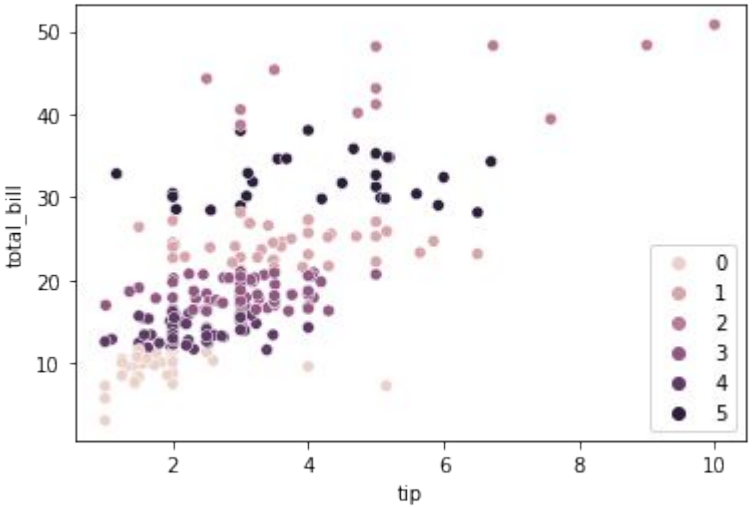



```
cls= AgglomerativeClustering().fit(X)
```

Model prediction size



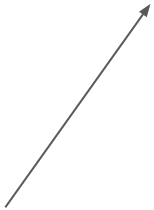
original data with labels (size)




```
kmeans = KMeans(n_clusters=6, random_state=0,
n_init="auto").fit(X)
```

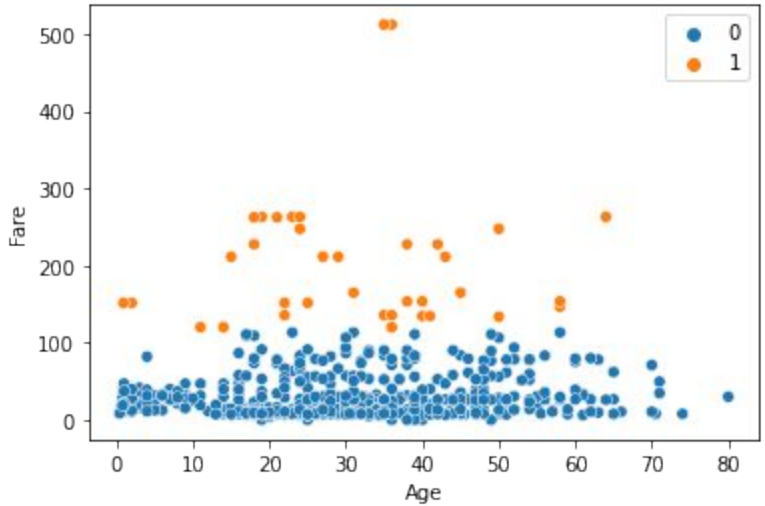
Titanic

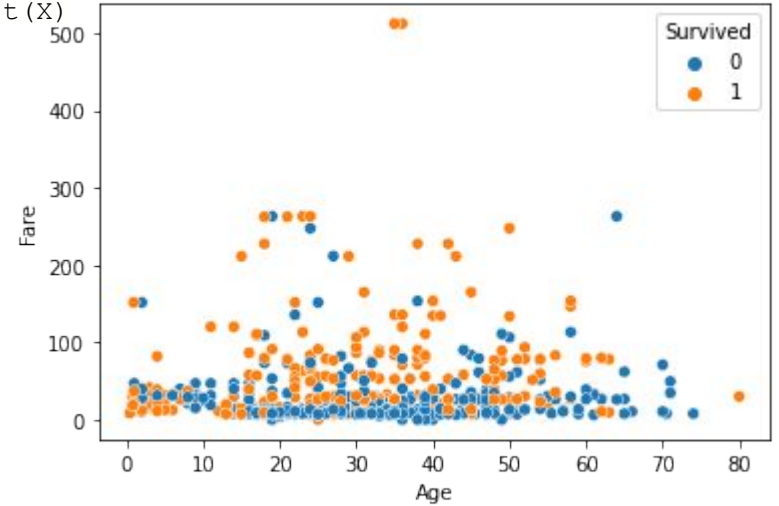
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C



Titanic model pipeline

```
titanic = pd.read_csv('titanic.csv')
X = titanic[['Fare', 'Age']]
X.fillna(0, inplace=True)
X = X.to_numpy()
y = titanic[['Survived']].to_numpy()

clf = KMeans(n_clusters=2, random_state=0, n_init="auto").fit(X)

correct = (clf.predict(X) == y.squeeze()).sum()

percent = correct/(len(X)) # 0.64
```





Mnist model

```
from numpy import genfromtxt

mnist_dataset = genfromtxt('/content/sample_data/mnist_test.csv', delimiter=',')
X = mnist_dataset[0:100][1:785] #100 obrazków z mnista

clf = KMeans(n_clusters=10, random_state=0, n_init="auto").fit(X)

clf.predict(X)
array([0, 4, 9, 8, 4, 6, 3, 8, 5, 7, 7, 2, 9, 4, 6, 8, 2, 6, 3, 5, 0, 8,
       0, 3, 7, 3, 8, 9, 4, 0, 4, 0, 9, 5, 6, 2, 4, 0, 4, 4, 2, 1, 4, 4,
       0, 4, 4, 1, 8, 0, 6, 0, 6, 8, 9, 8, 4, 8, 4, 3, 5, 3, 4, 5, 3, 4,
       8, 0, 7, 2, 7, 0, 5, 4, 4, 0, 2, 4, 5, 3, 0, 5, 3, 1, 1, 5, 0, 3,
       4, 0, 0, 5, 0, 4, 1, 4, 4, 0, 8], dtype=int32)
```

References

- https://en.wikipedia.org/wiki/K-means_clustering
- https://en.wikipedia.org/wiki/Hierarchical_clustering