# Metric spaces, Voronoi tessellation, and vector quantization

## Pre-lecture 6 video

A **metric** on a set $X$ is a function (called *distance function* or simply **distance**)

$$d : X \times X \to [0, \infty),$$

where $[0, \infty)$ is the set of non-negative real numbers and for all $x, y, z \in X$, the following three axioms are satisfied:

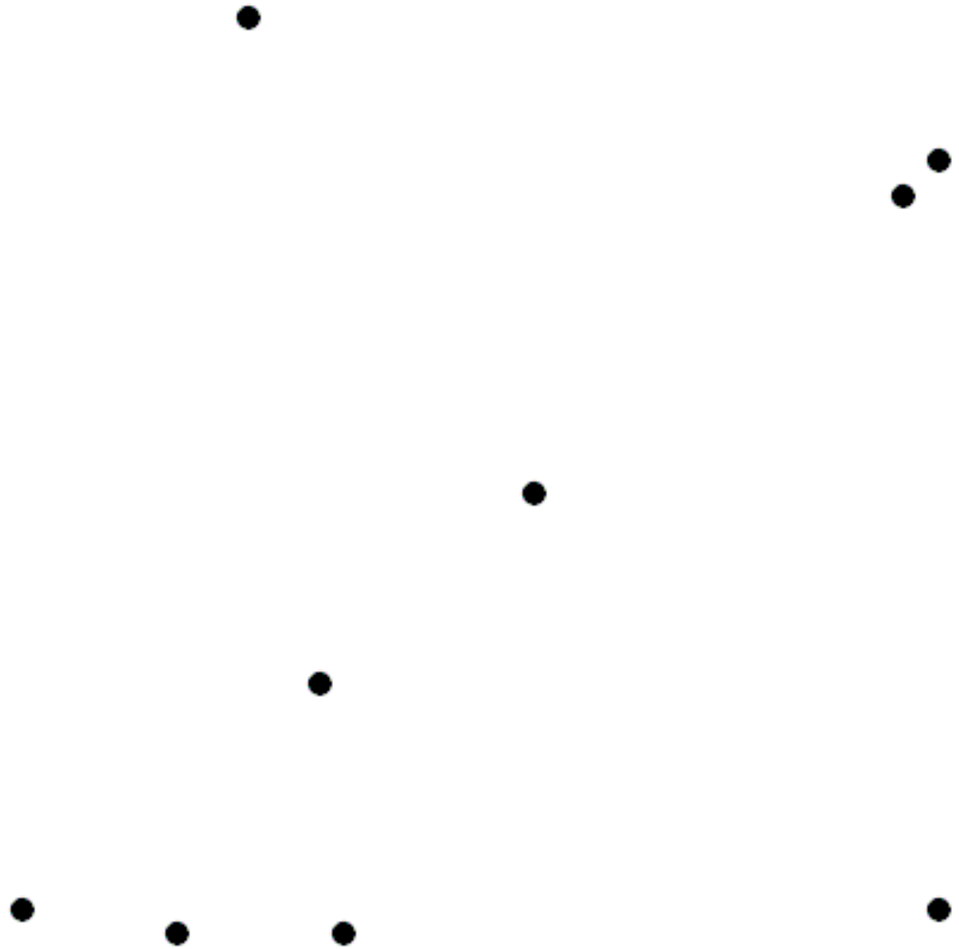
1. $d(x, y) = 0 \Leftrightarrow x = y$ identity of indiscernibles
2. $d(x, y) = d(y, x)$ symmetry
3. $d(x, y) \leq d(x, z) + d(z, y)$ subadditivity or triangle inequality

ML is full of metric spaces!

- Euclidean (L2)
- Manhattan (L1)
- Minkowski (Lp norm for arbitrary p)
- Mahalanobis (generalization of z-score)
- Hamming (binary feature distance)

# Voronoi diagram

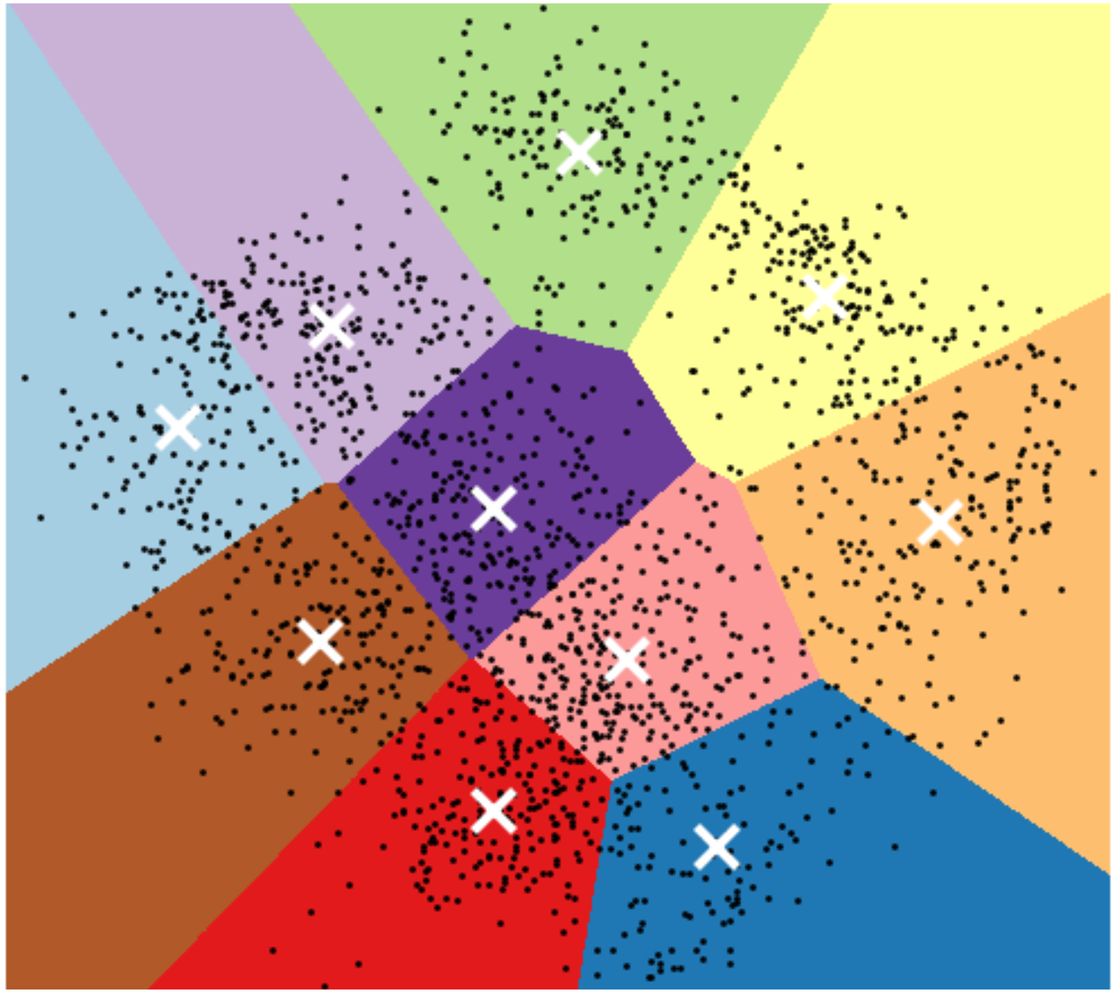
- Given a set of points $p$, the Voronoi region $R_k$ around a point $p_k$ consists of every location that is closer to $p_k$ than $p_j, \forall j \neq k$
- Draw a line from $p_j$ to $p_k$, the locations closer to $p_k$ lie on its side of the perpendicular bisecting line
- $R_k$ is the closed region defined by the intersection of all the perpendicular bisectors between $p_k$ and all its neighboring points $p_j, \forall j \neq k$
- Requires a definition of distance: a metric space!



# K-means

## Clustering 101

- Assume that the data is described by a set of convex clusters $C = \{C_1, \ldots C_k\}$
- Each cluster has different mean but equal variance
- Every datapoint is assumed to be generated by the closest mean (aka Voronoi tessellation)



# K-means

## Algorithm

Choose centroids such that we $\arg\min_{C} \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_{C_i}\|^2$ where

$\mu_{C_i} = \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} x_j$. Here $\|\cdot\|$ indicates whichever metric space we are using

(almost always the Euclidean norm, aka L2) and $|\cdot|$ indicates set cardinality which in this case means the number of data points belonging to that particular cluster

# K-means

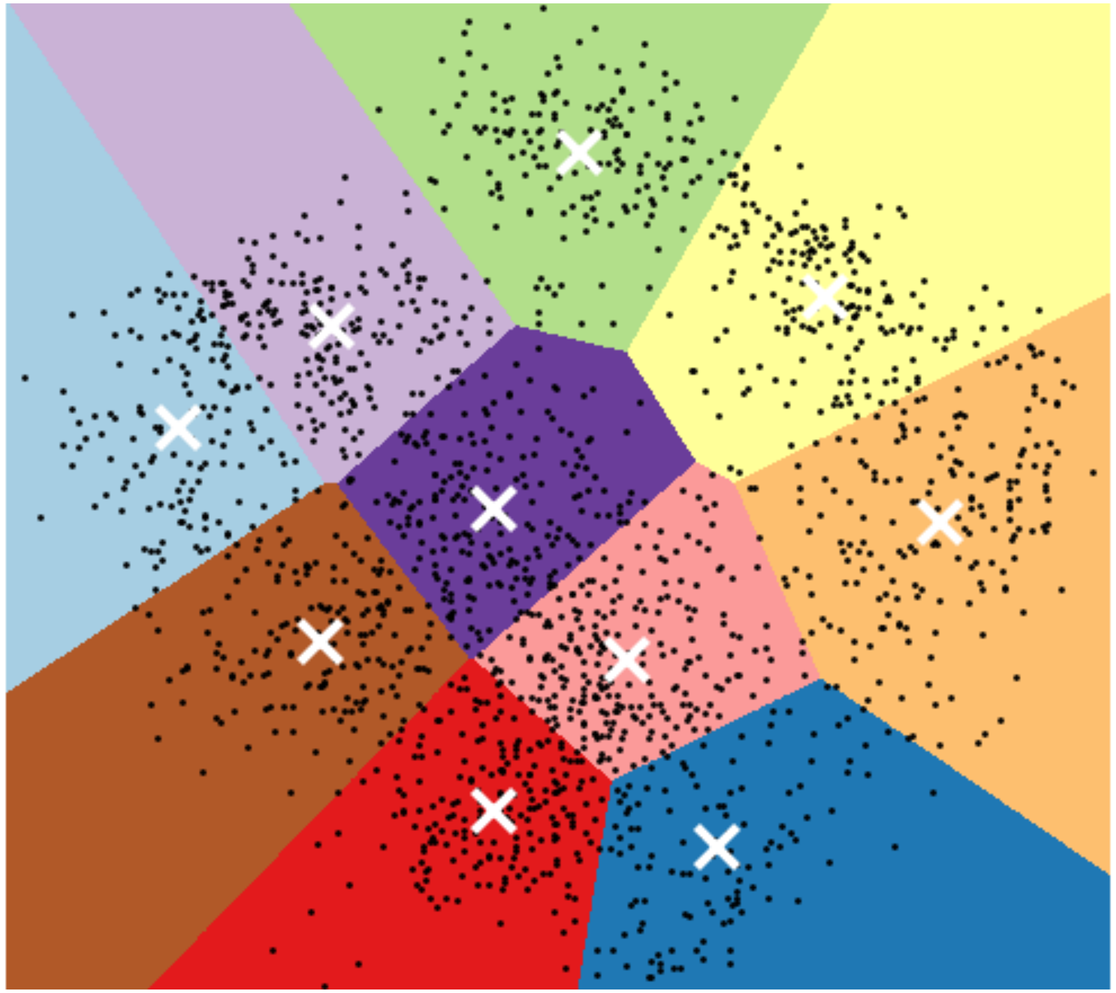
## Simple batch implementation

- Initialize the centroids
  - e.g. pick k of the datapoints at random
- Do
  - Assign each datapoint to its closest cluster mean
  - Calculate the new mean of each cluster given the cluster assignments
  - Until the total distance the k-means moved from the last iteration is less than some threshold

# Vector quantization

## Clustering 102

- Assume that the data is described by a set of convex clusters $C = \{C_1, \ldots C_k\}$, each of which can be reduced to a prototype vector (aka the centroid)
- Data can be compressed or quantized or categorized simply by throwing away the data itself and representing the data as the centroid prototype
- Objective function we are minimizing is the same as K-means
- Implementation can be online:
  - Pick a datapoint at random, find the closest prototype vector $\mu_\mathbf{i}$
  - Move the prototype vector fractionally towards the datapoint $\mu_i^{t+1} = \mu_\mathbf{i}^\mathbf{t} + \eta(\mathbf{x} - \mu_\mathbf{i}^\mathbf{t}), \eta < 1$



# K-means etc

**Lecture 06**

A reading list for this week:

Bishop 9.1,9.2,

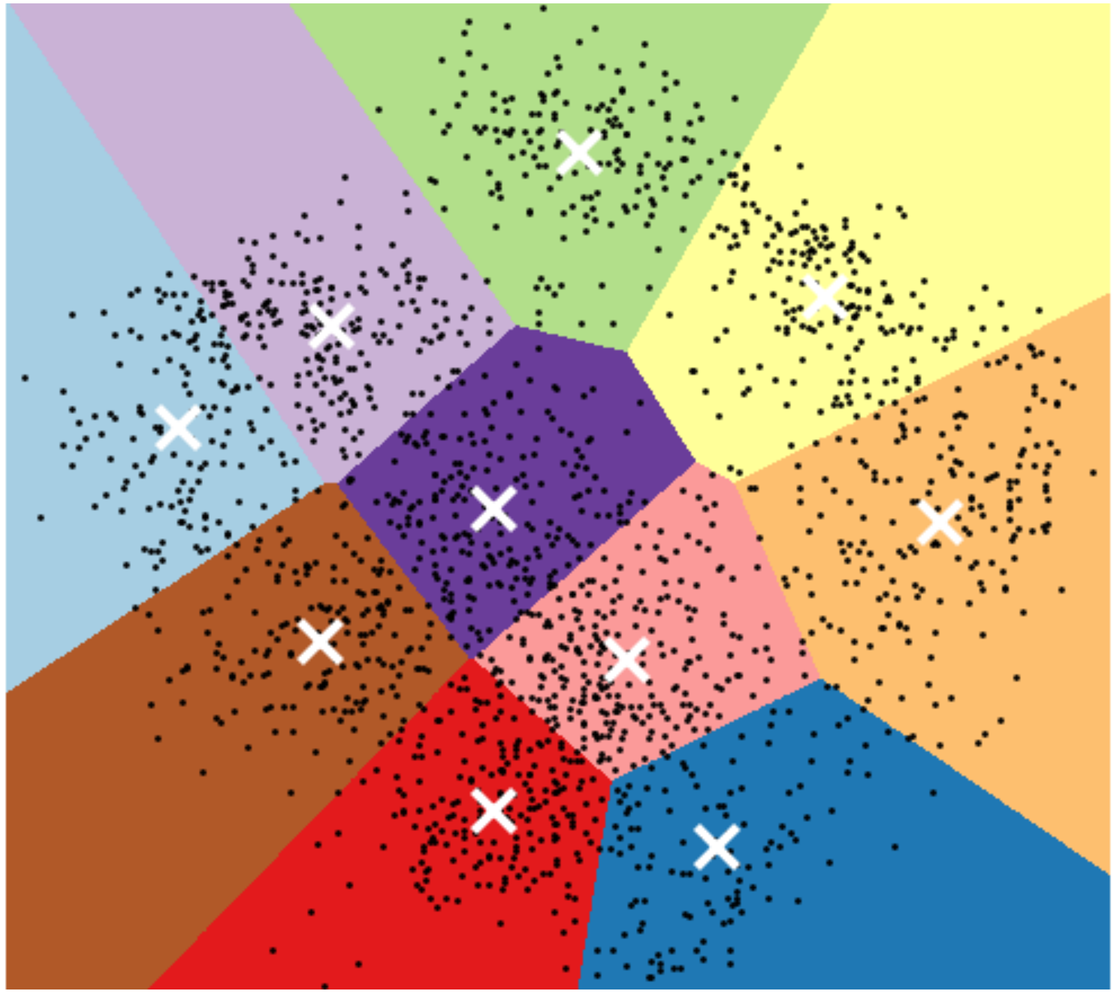
Hastie et al 13.1, 13.2

Bonaccorso “Clustering fundamentals”

# K-means

## Clustering 101

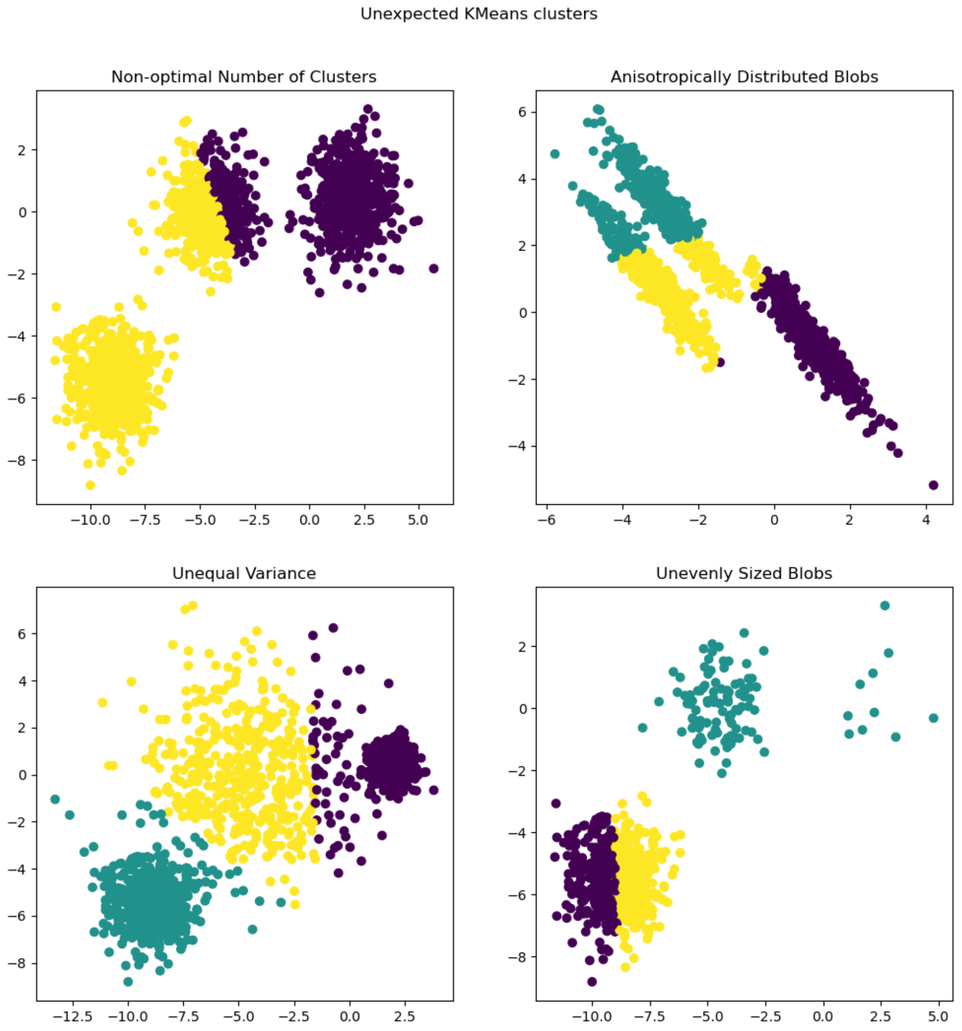
- Assume that the data is described by a set of convex clusters $C = \{C_1, \ldots C_k\}$
- Each cluster has different mean but equal variance
- Every datapoint is assumed to be generated by the closest mean (aka Voronoi tessellation)



# K-means

## Clustering 101

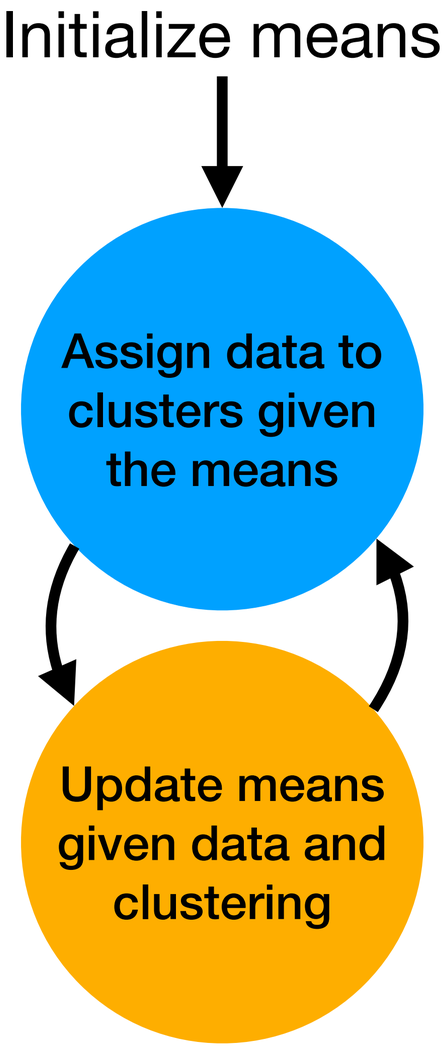
- Assume that the data is described by a set of convex clusters $C = \{C_1, \ldots C_k\}$
- Each cluster has different mean but equal variance
- Every datapoint is assumed to be generated by the closest mean (aka Voronoi tessellation)



# K-means

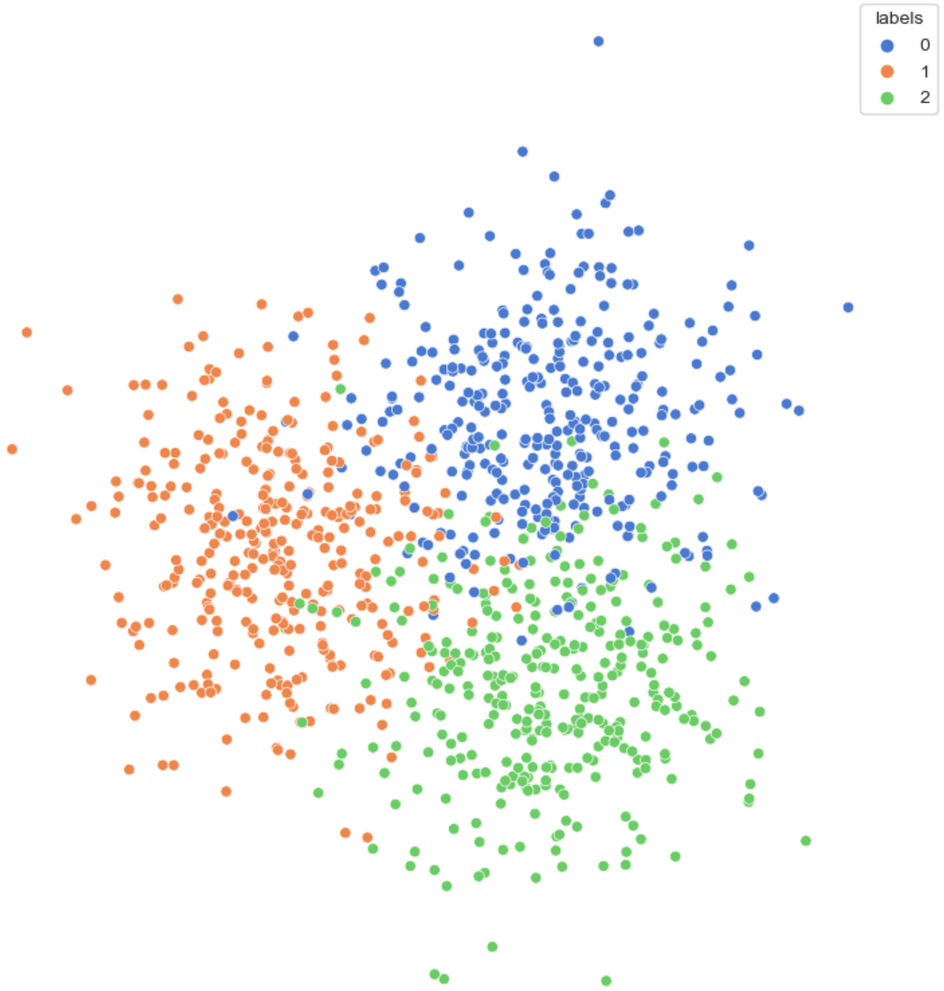
## Simple implementation

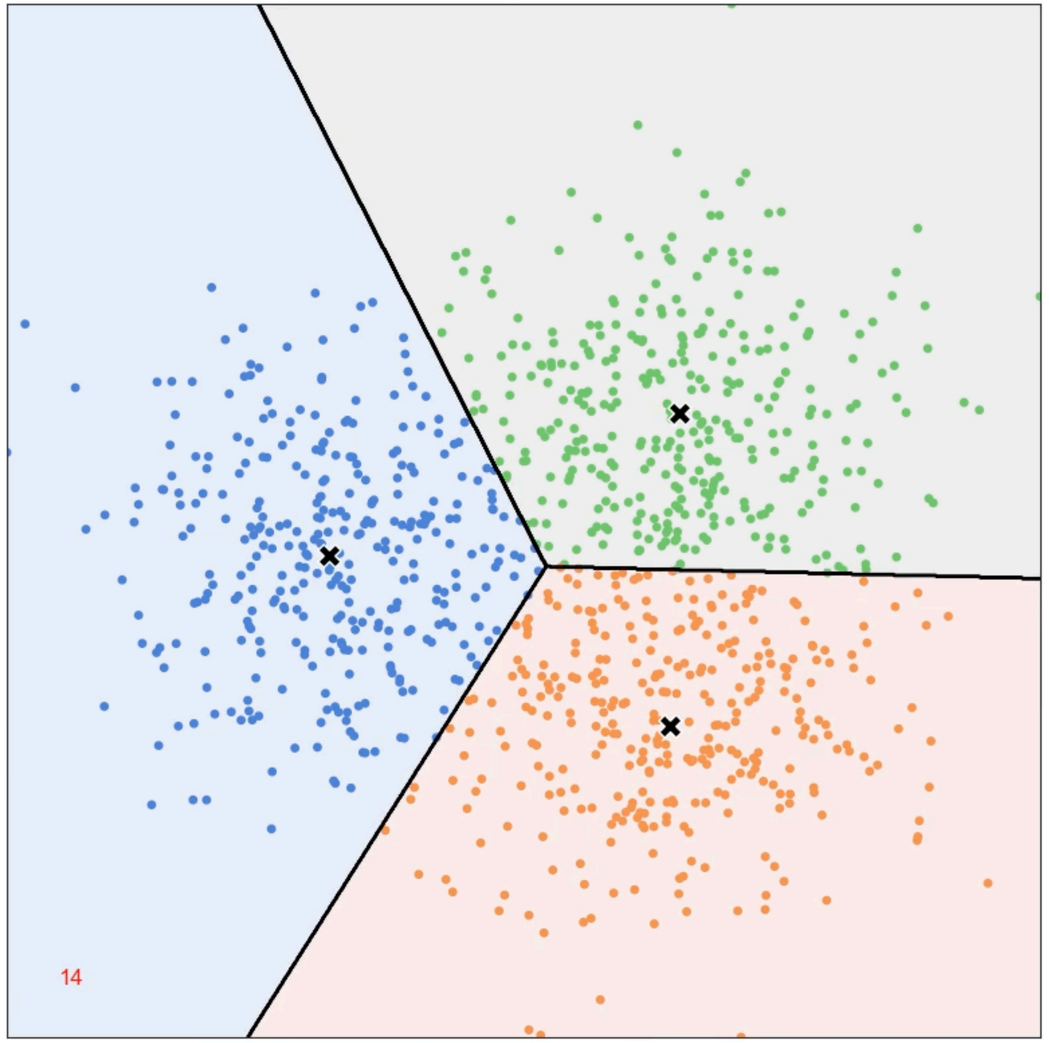
- Initialize the centroids
  - e.g. pick k of the datapoints at random
- Do
  - Assign each datapoint to its closest cluster mean
  - Calculate the new mean of each cluster given the cluster assignments
  - Until the total distance the k-means moved from the last iteration is less than some threshold



# K-means

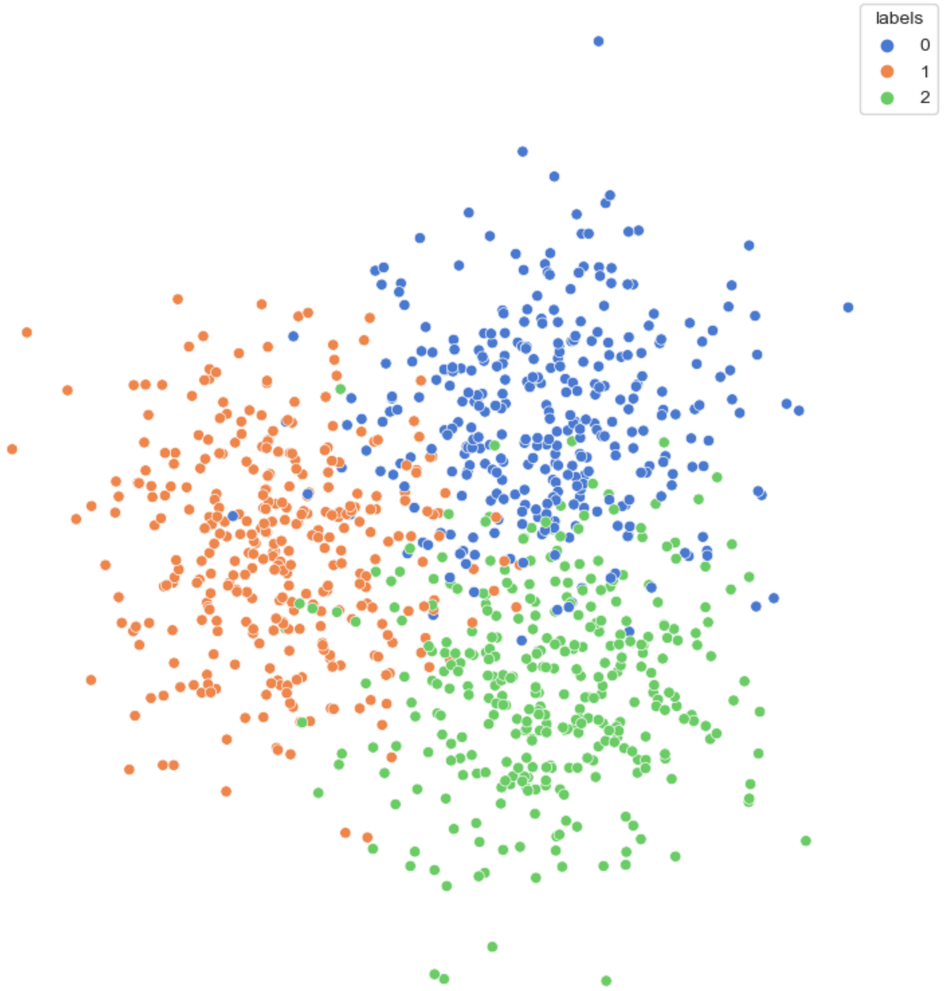
## Watch it work in the best case scenario

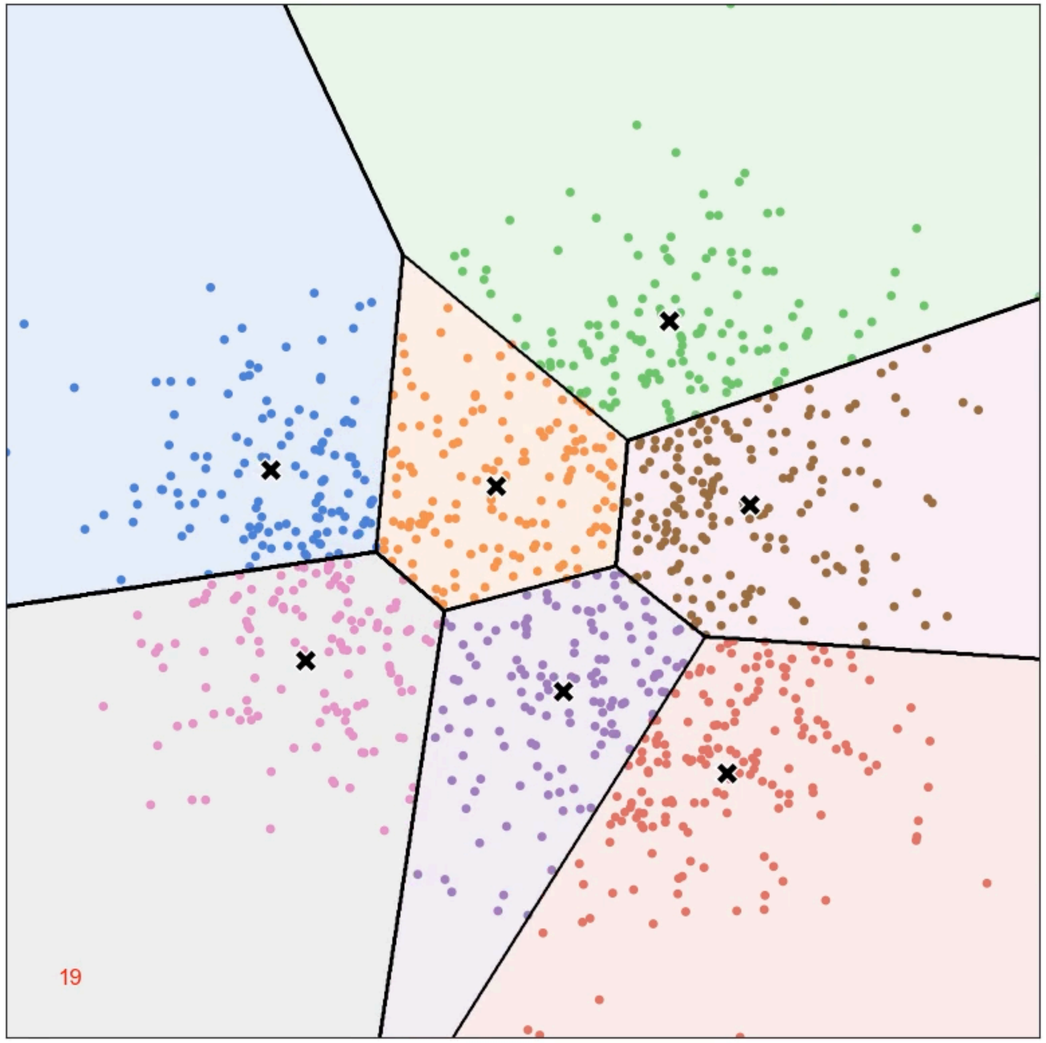


# K-means

## Different k, same data



# Measuring clustering performance

## Without ground truth labels

- **a**: The mean distance between a sample and all other points in the same class.
- **b**: The mean distance between a sample and all other points in the *next nearest cluster*.

The Silhouette Coefficient *s* for a single sample is then given as:

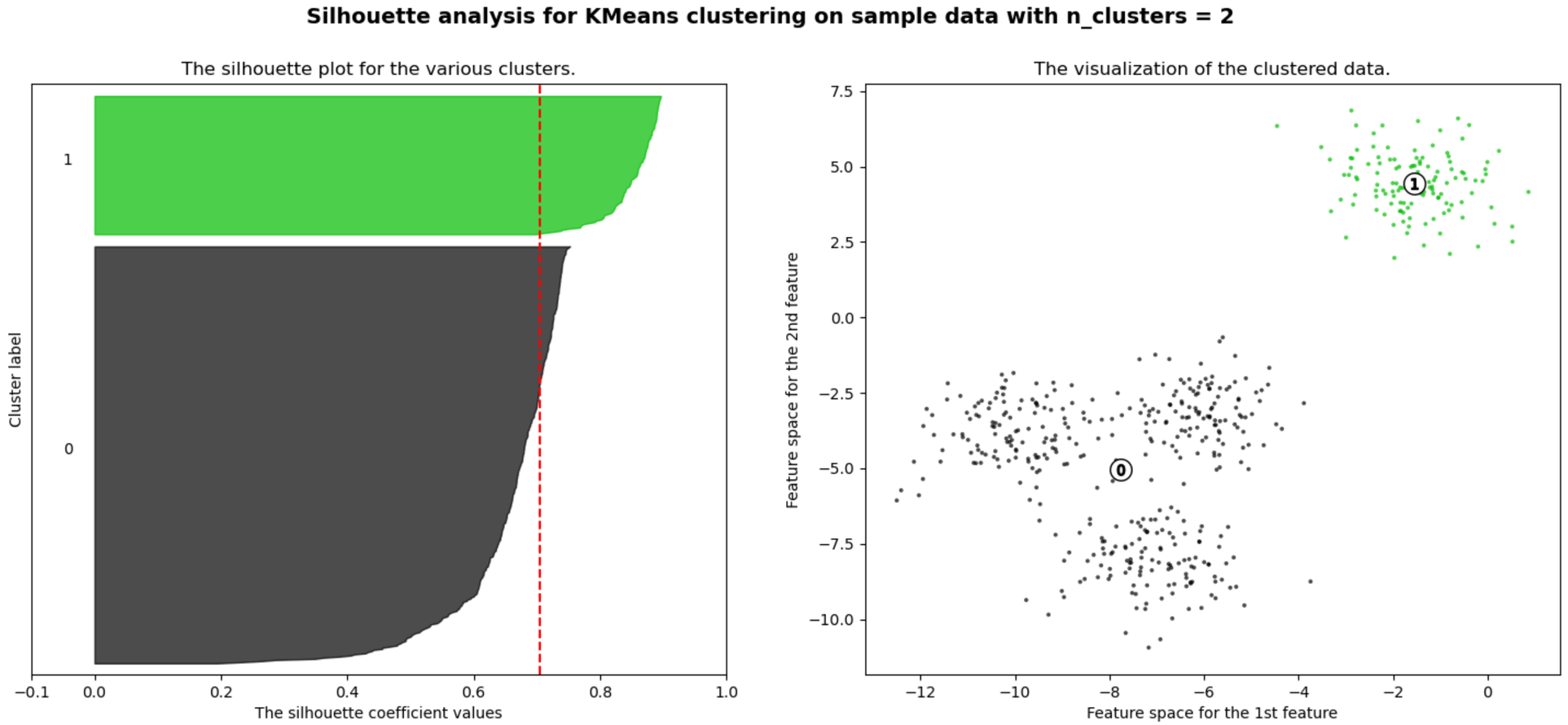
$$s = \frac{b - a}{max(a, b)}$$

The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient for each sample.

- The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters.
- The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.
- Gives worse results with non-convex clusters

# Model selection of k

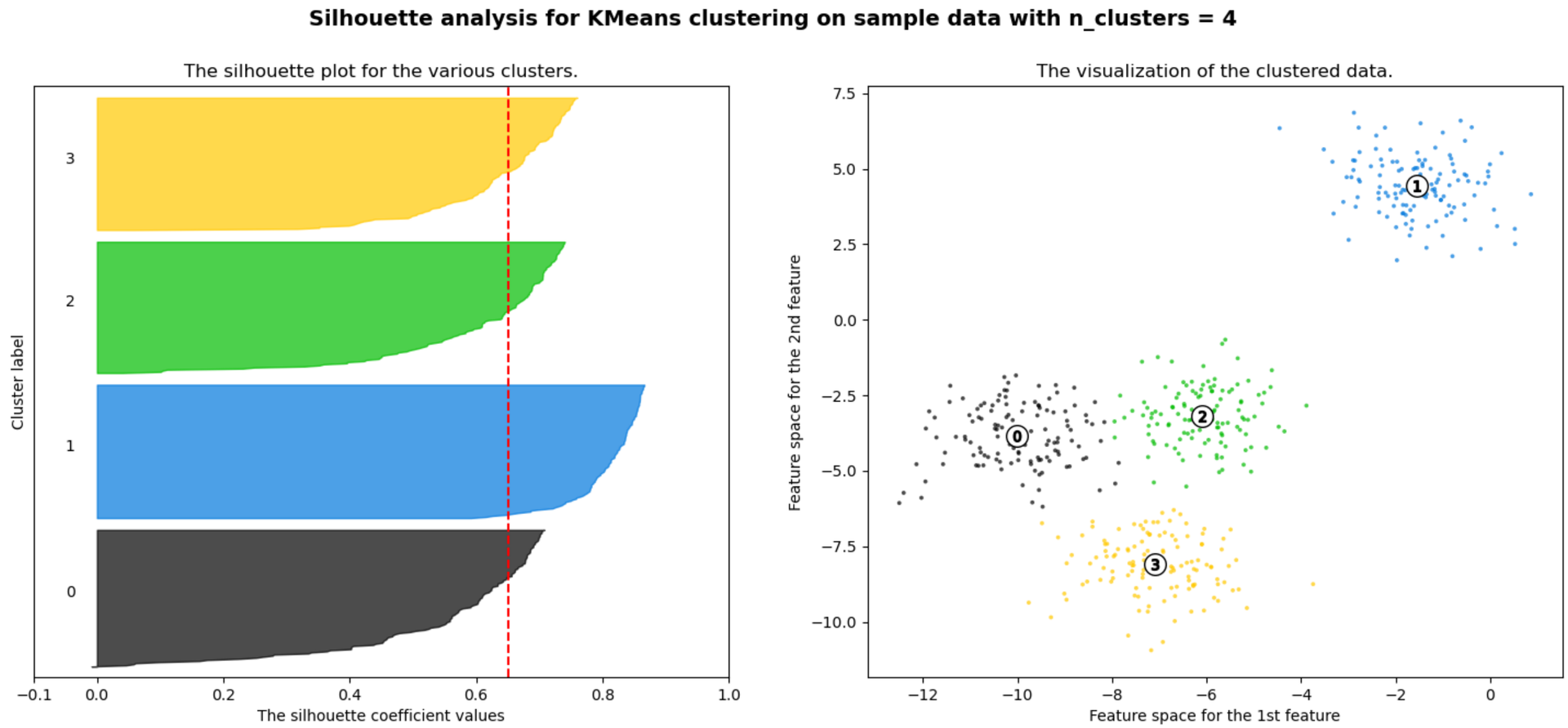
## With no labels




- See demo at https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html#sphx-glr-auto-examples-cluster-plot-kmeans-silhouette-analysis-py

# Model selection of k

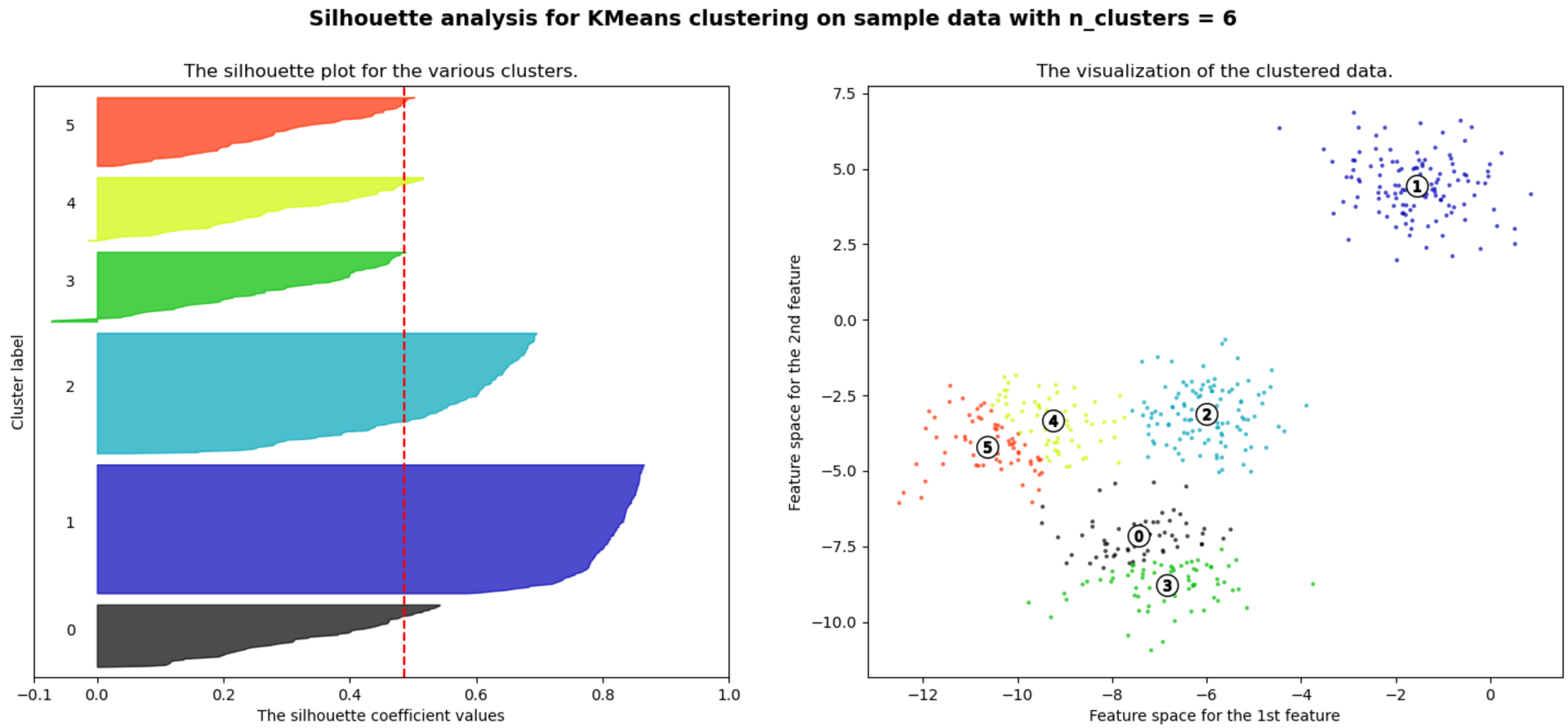
## With no labels




- See demo at https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html#sphx-glr-auto-examples-cluster-plot-kmeans-silhouette-analysis-py

# Model selection of k

## With no labels




- See demo at https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html#sphx-glr-auto-examples-cluster-plot-kmeans-silhouette-analysis-py

# Model selection of k

## With no labels

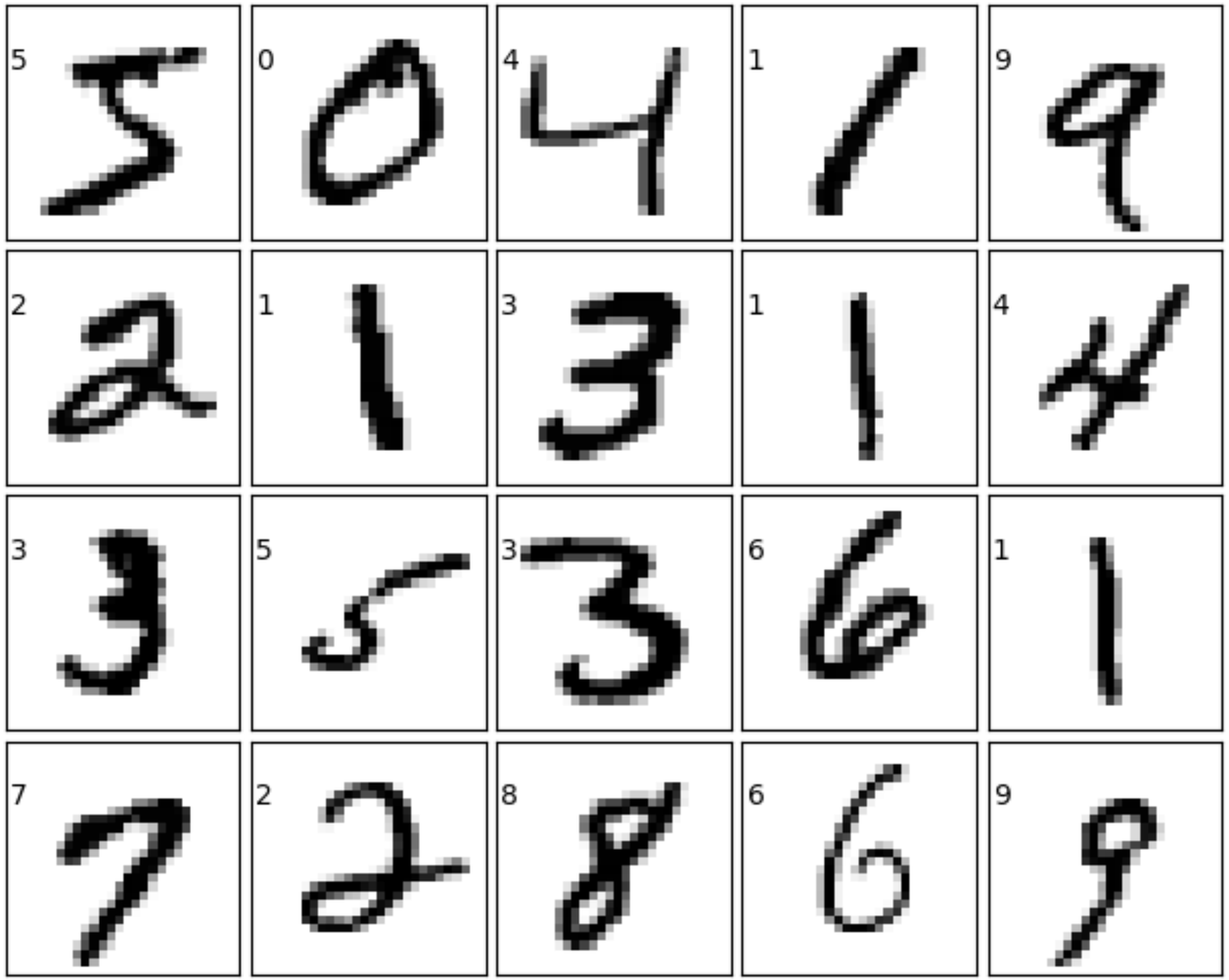
Out:

```
For n_clusters = 2 The average silhouette_score is : 0.7049787496083262
For n_clusters = 3 The average silhouette_score is : 0.5882004012129721
For n_clusters = 4 The average silhouette_score is : 0.6505186632729437
For n_clusters = 5 The average silhouette_score is : 0.561464362648773
For n_clusters = 6 The average silhouette_score is : 0.4857596147013469
```

- See demo at https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html#sphx-glr-auto-examples-cluster-plot-kmeans-silhouette-analysis-py

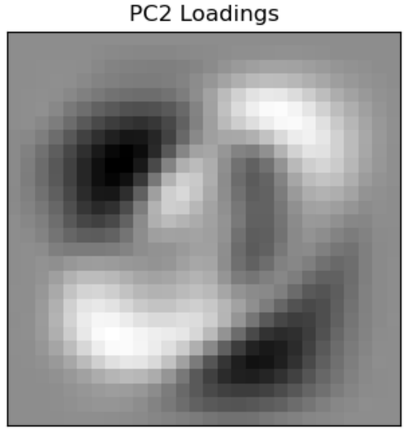
# Clustering != categorization

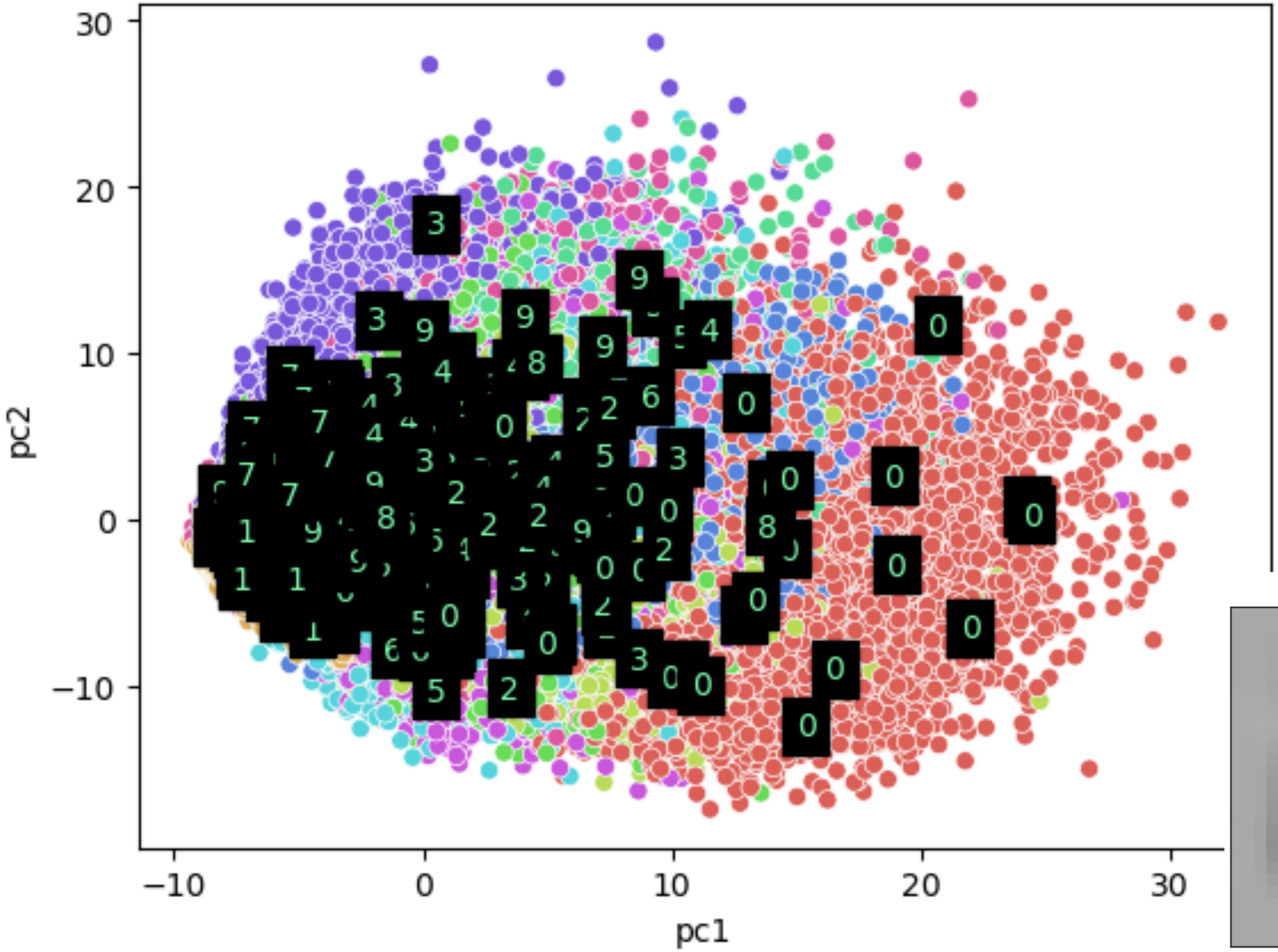
## MNIST 2D PCA

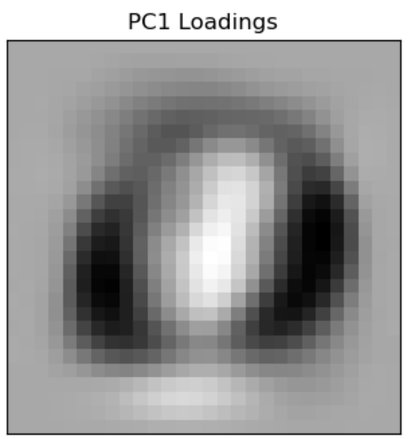


# Clustering != categorization

## MNIST 2D PCA

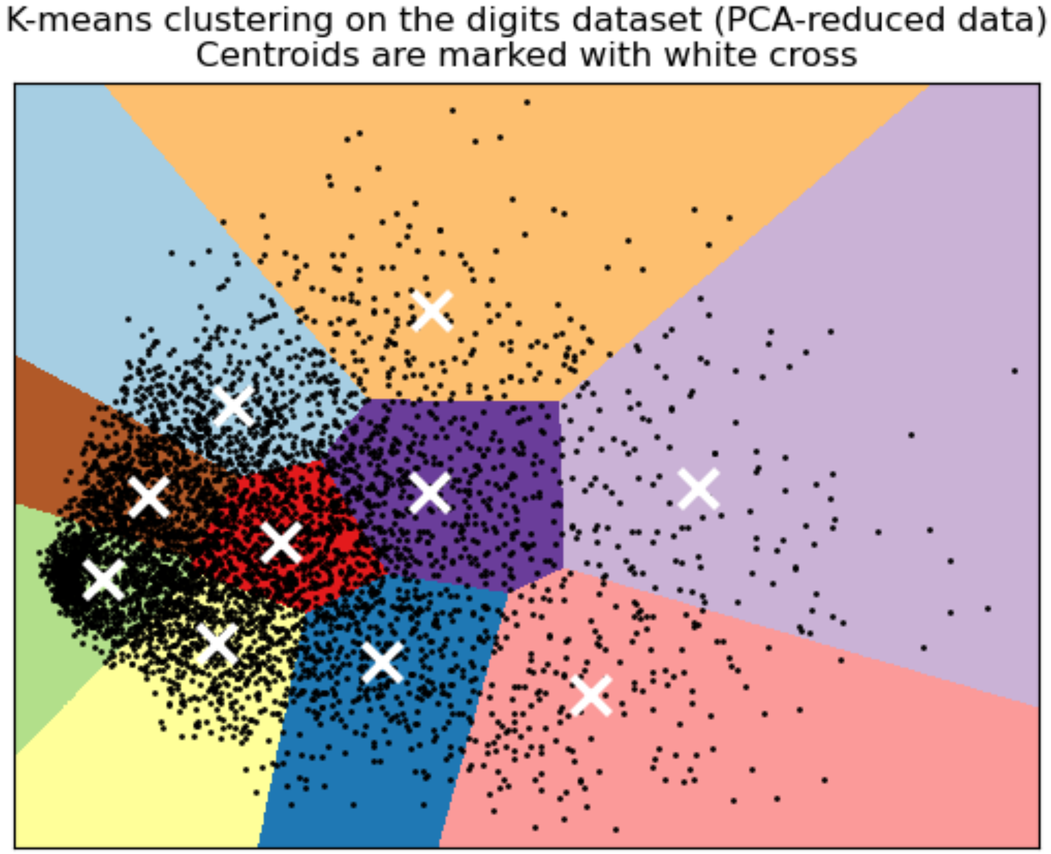


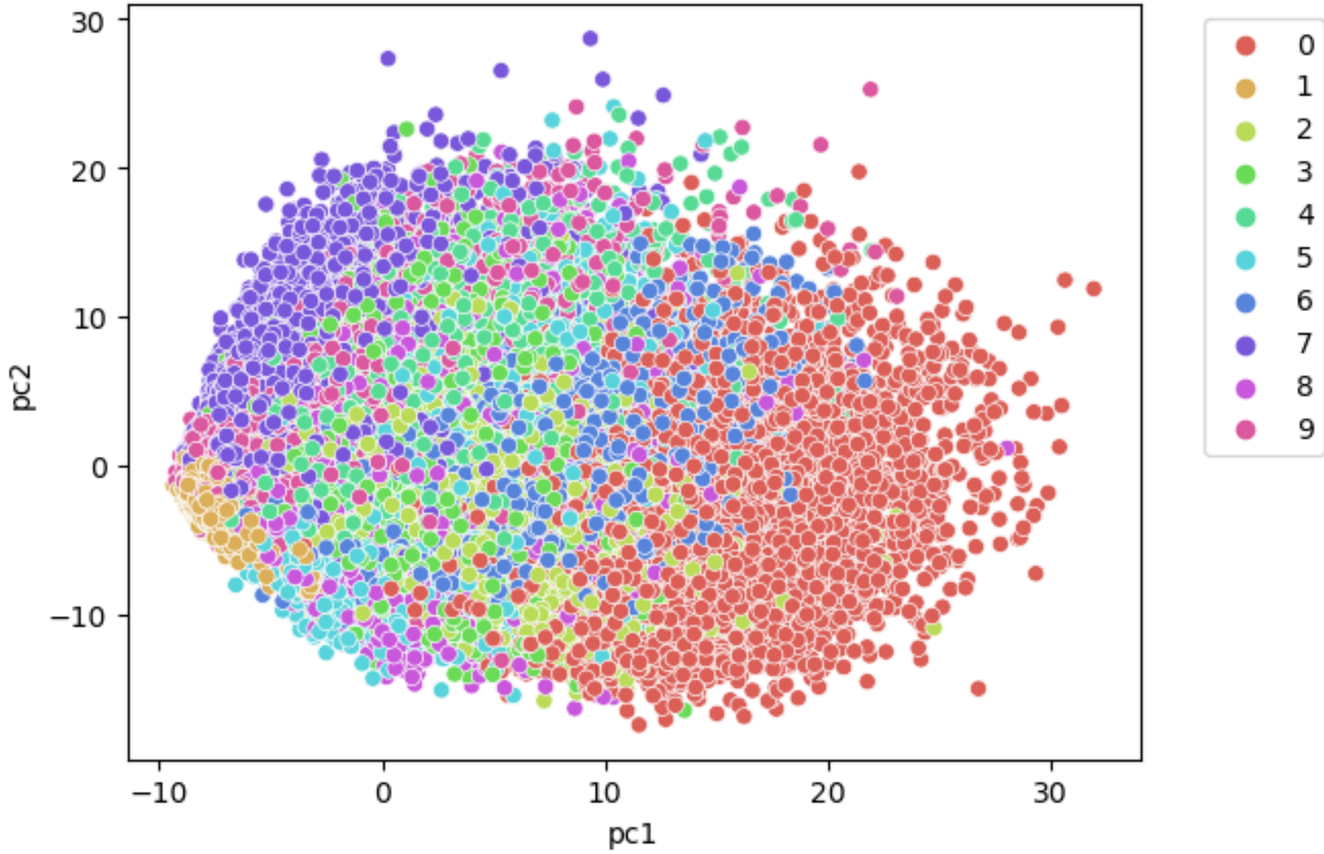



# Clustering != categorization

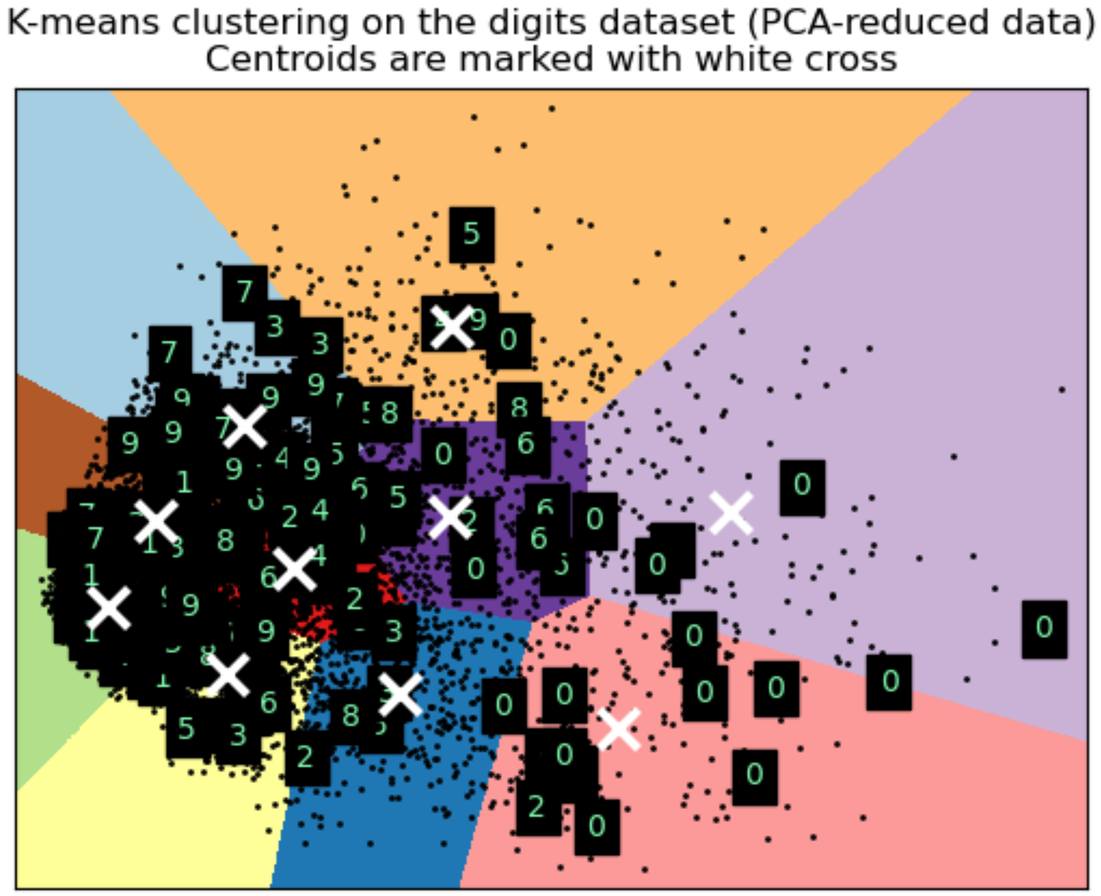
## MNIST 2D PCA

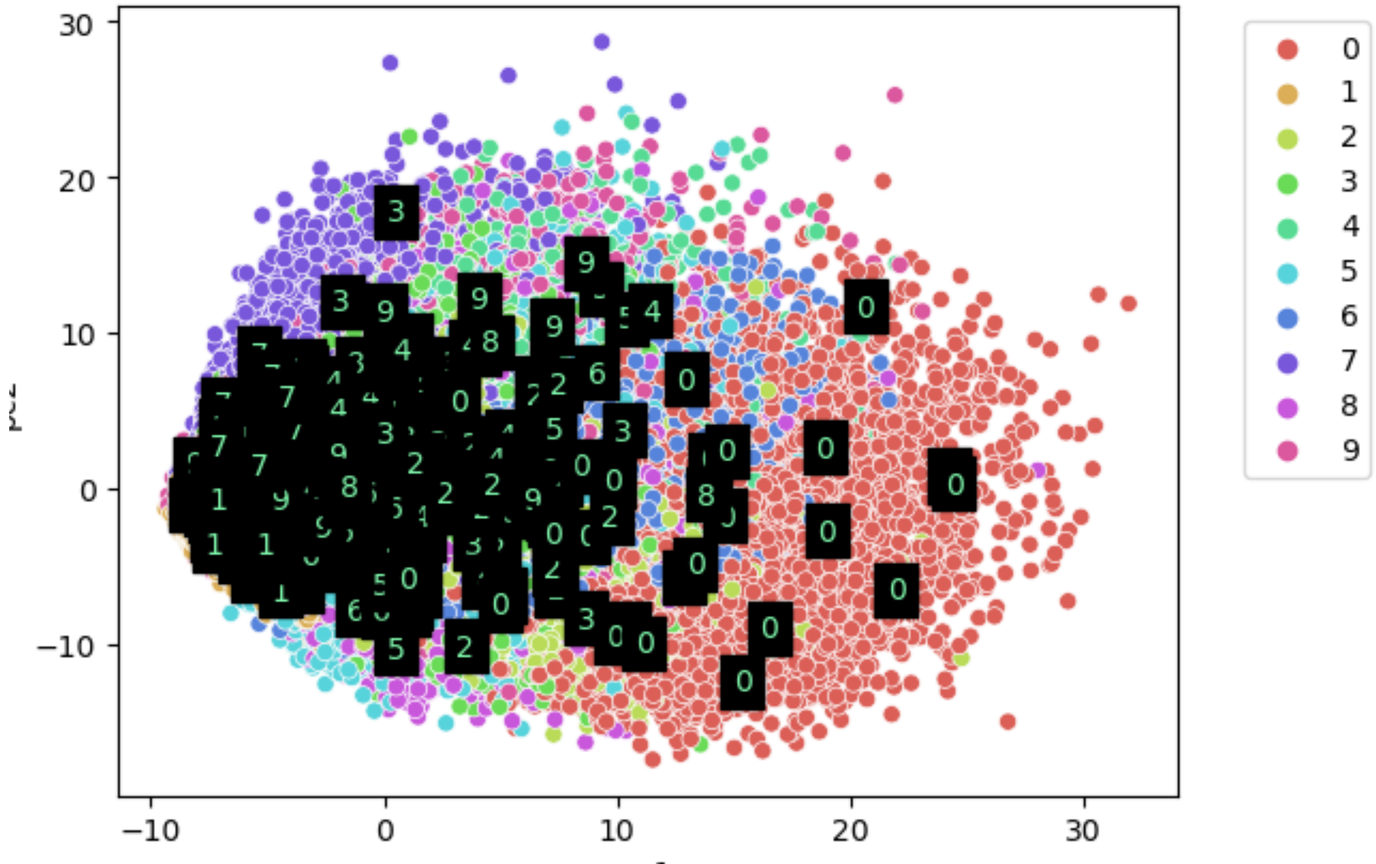


# Clustering != categorization

## MNIST 2D PCA



# K-means as Vector Quantization

https://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html

Original image (96,615 colors)

Quantized image (64 colors, K-Means)

3 bytes / pixel = 16M colors

1 byte / pixel = 256 colors