CITS5508 Machine Learning

Débora Corrêa (Unit Coordinator and Lecturer)
UWA

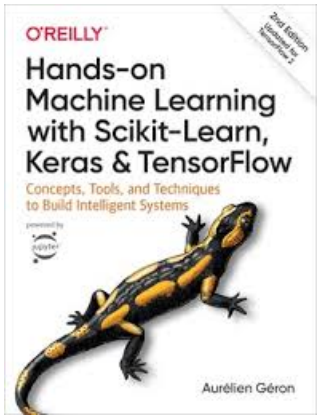
2023

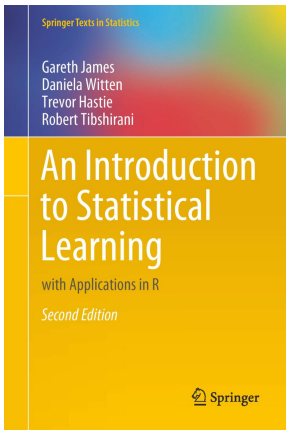
Today

Quick review on PCA.
Clustering techniques: K-means.

Hands-on Machine Learning with Scikit-Learn & TensorFlow (chapter 9)

An Introduction to Statistical Learning (chapter 12)



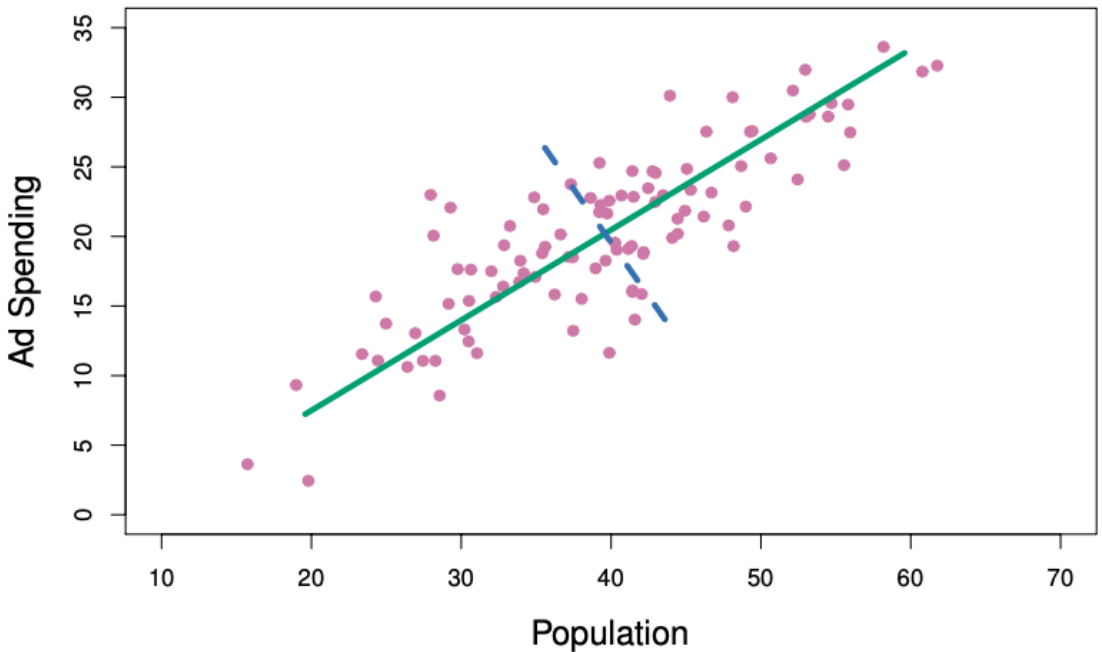


Principal Component Analysis (PCA)

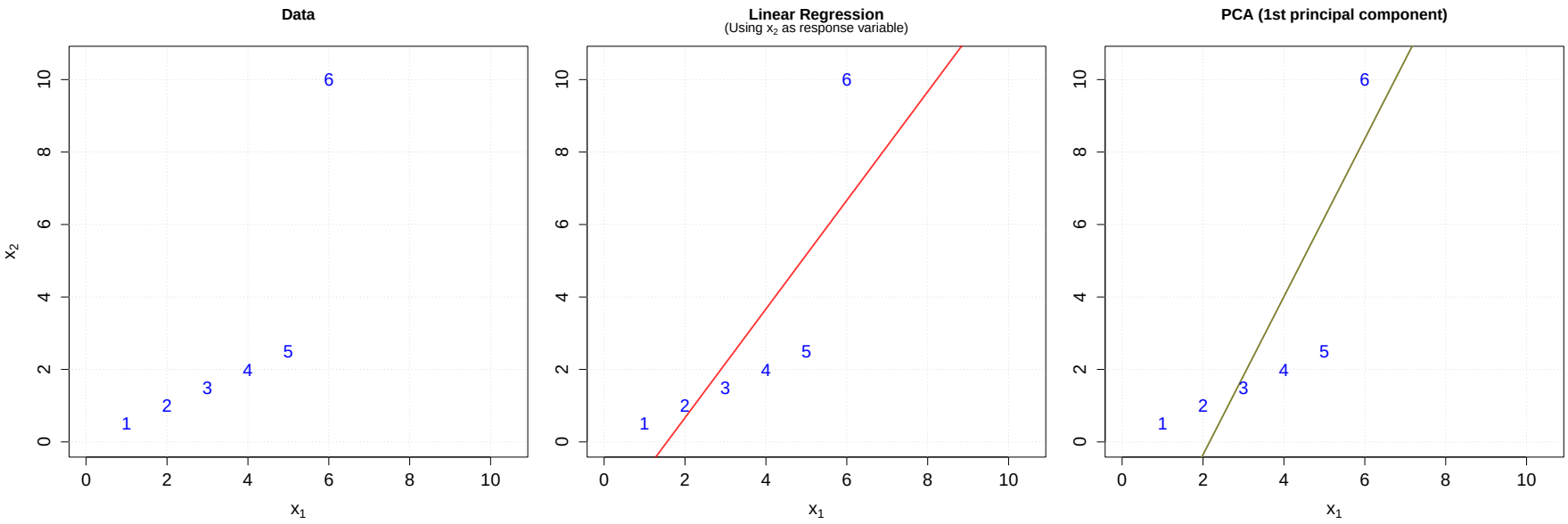
The first principal component direction of the data is the direction along which the instances vary the most (the resulting projected instances have the largest possible variance).

direction where it vary most

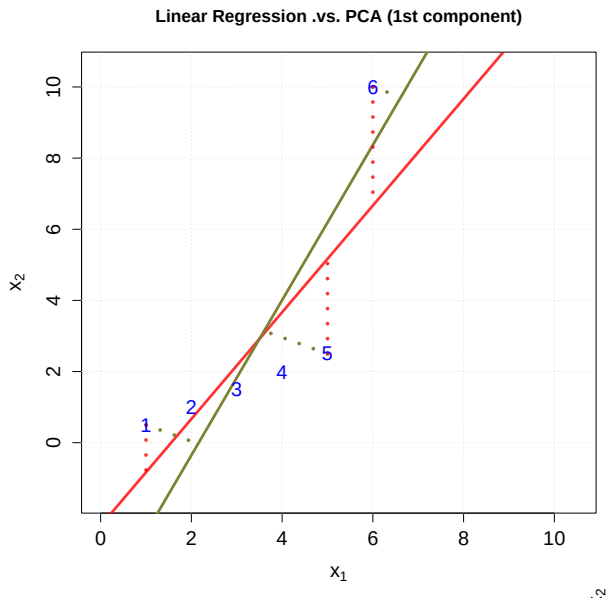
The first principal component direction also defines the line that minimises the sum of the squared *perpendicular* distances between each point and the line.

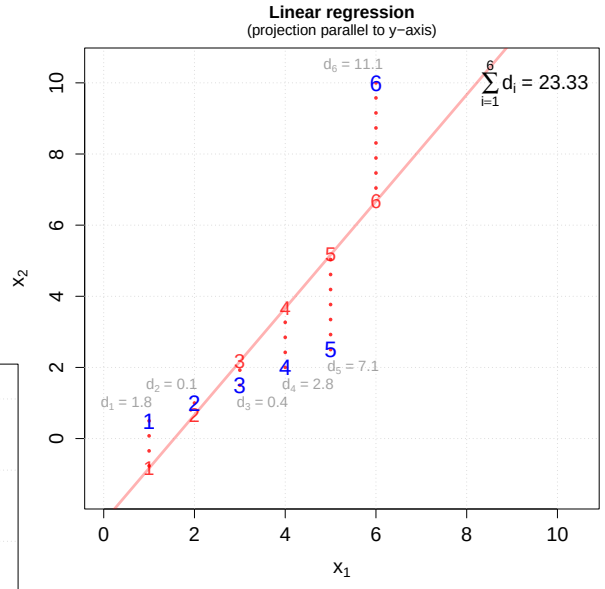


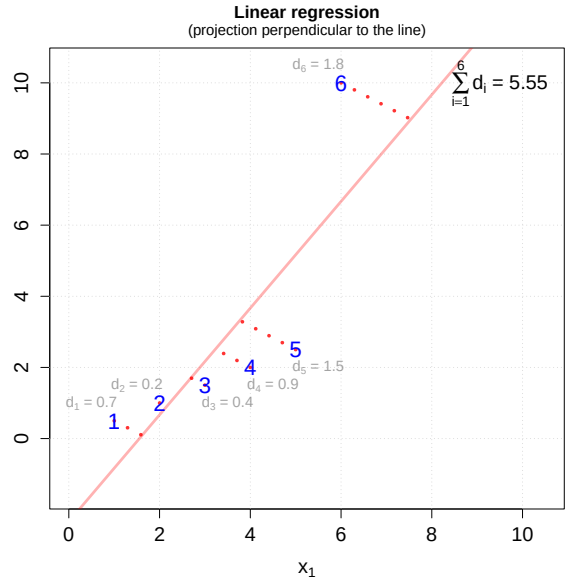
Principal Component Analysis (PCA)

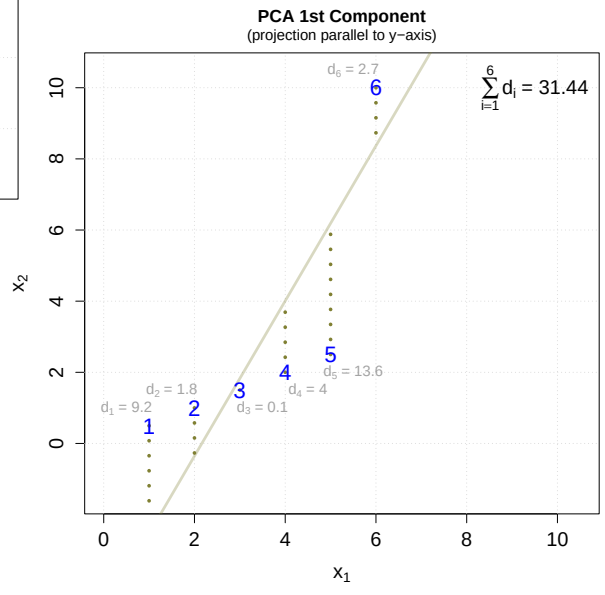


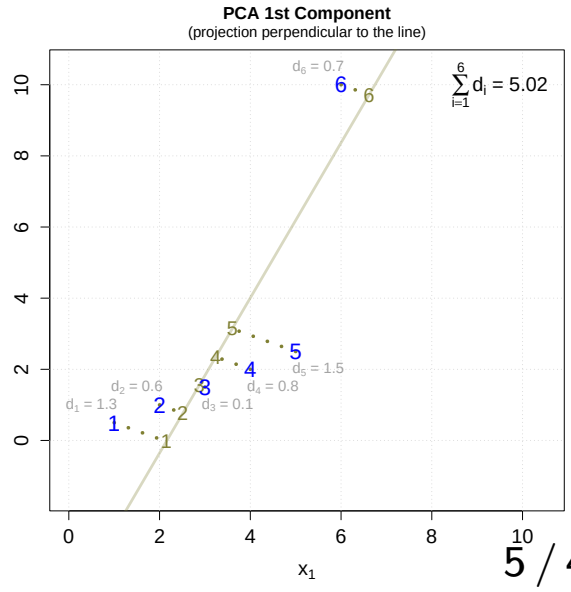
Principal Component Analysis (PCA)





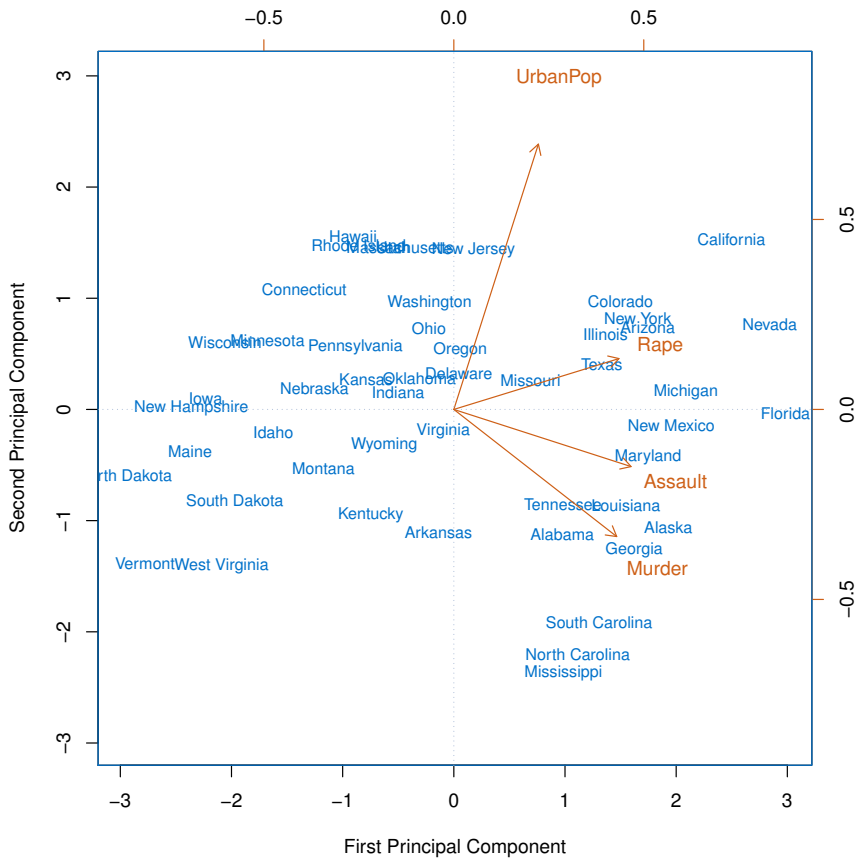






PCA - Example

USArrests data: For each of the 50 states in the United States, it contains the number of arrests per $100,000$ residents for each of three crimes (`Assault`, `Murder`, and `Rape`) and the percent of the population in each state living in urban areas (`UrbanPop`).



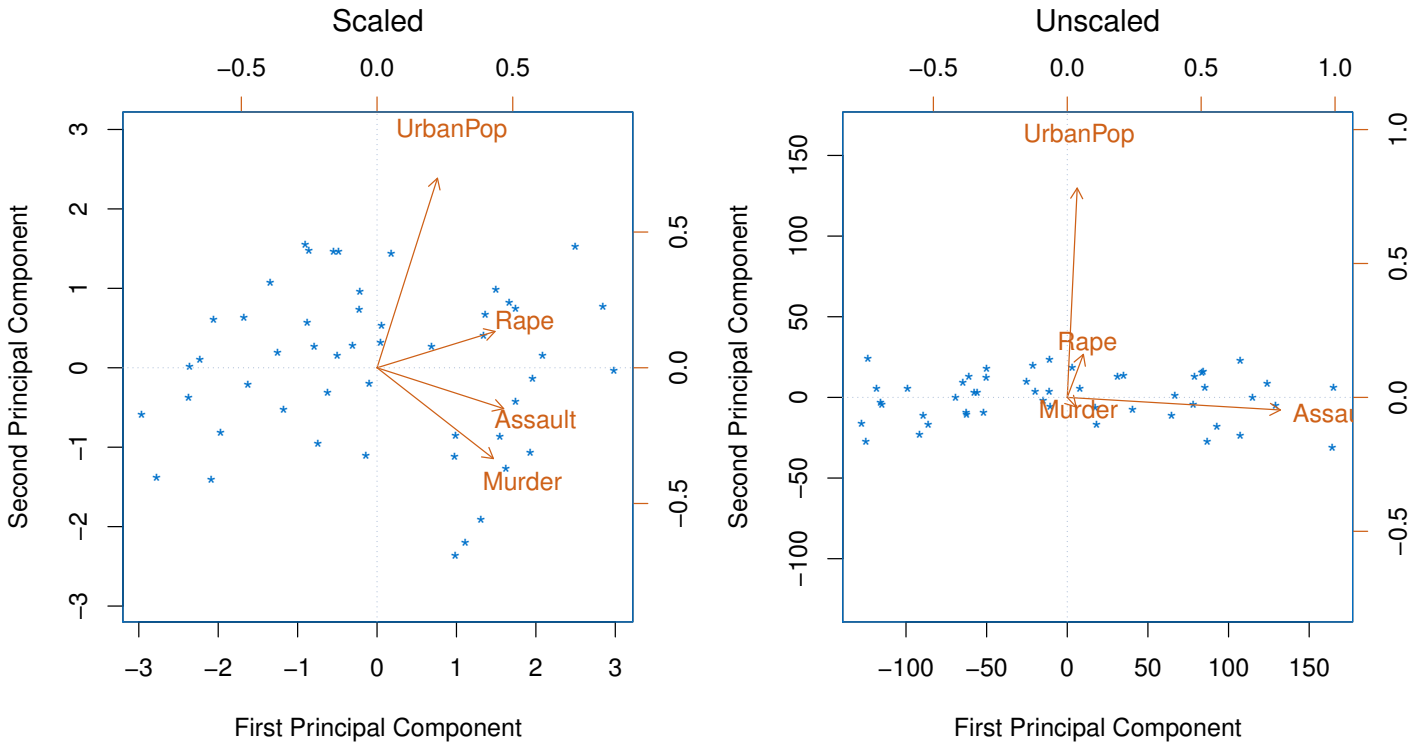
	PC1	PC2
Murder	0.5358	-0.4181
Assault	0.5831	-0.1879
UrbanPop	0.2781	0.8728
Rape	0.5434	0.1673

PCA on Unscaled Data

The variables are measured in different units: Murder, Rape, and Assault are recorded as the number of occurrences per $100,000$ people, and UrbanPop is the percentage of the state's population living in an urban area.

They have variance of 18.97, 87.73, 6945.16, and 209.5, respectively.

PCA on Unscaled Data



Left: The variables scaled to have unit standard deviations. Right: The principal components using unscaled data.

Assault has the largest loading on the first principal component as it has the highest variance among the four variables. It is typically recommended to scale the variables to have unit standard deviation before applying PCA.

Examples of Unsupervised Tasks

`PCA`: The aim is to find a low-dimensional representation of the data set that contains most of its variance.

`Clustering`: The aim is to find homogeneous subgroups in the data set.

`Anomaly detection`: The aim is to represent the pattern of "normal" data, and use that to detect abnormal instances (often called *anomalies* or *outliers*).

`Density estimation`: The aim is to estimate the *probability density function* (PDF) of the random process that generated the data set (instances located in low-density regions are likely to be anomalies).

Examples Application Domains

- Fraud detection;
- Detection of defective products in manufacturing;
- Removal of outliers in a dataset before training a model;
- Customer or market segmentation;
- Recommender systems;
- Image segmentation;
- Search engines.

Example: A data set with many measurements (e.g. median household income, occupation, distance from the nearest urban area, etc.) for many individuals. We can use clustering for *market segmentation*: find the subgroups of people sharing most of the characteristics, and that might be more receptive to specific forms of advertising.

Clustering

Broad set of techniques that involve grouping instances (i.e. finding `subgroups` or `clusters`) in a data set based on similarities among the instances.

Aim: Partition the data set into distinct groups:

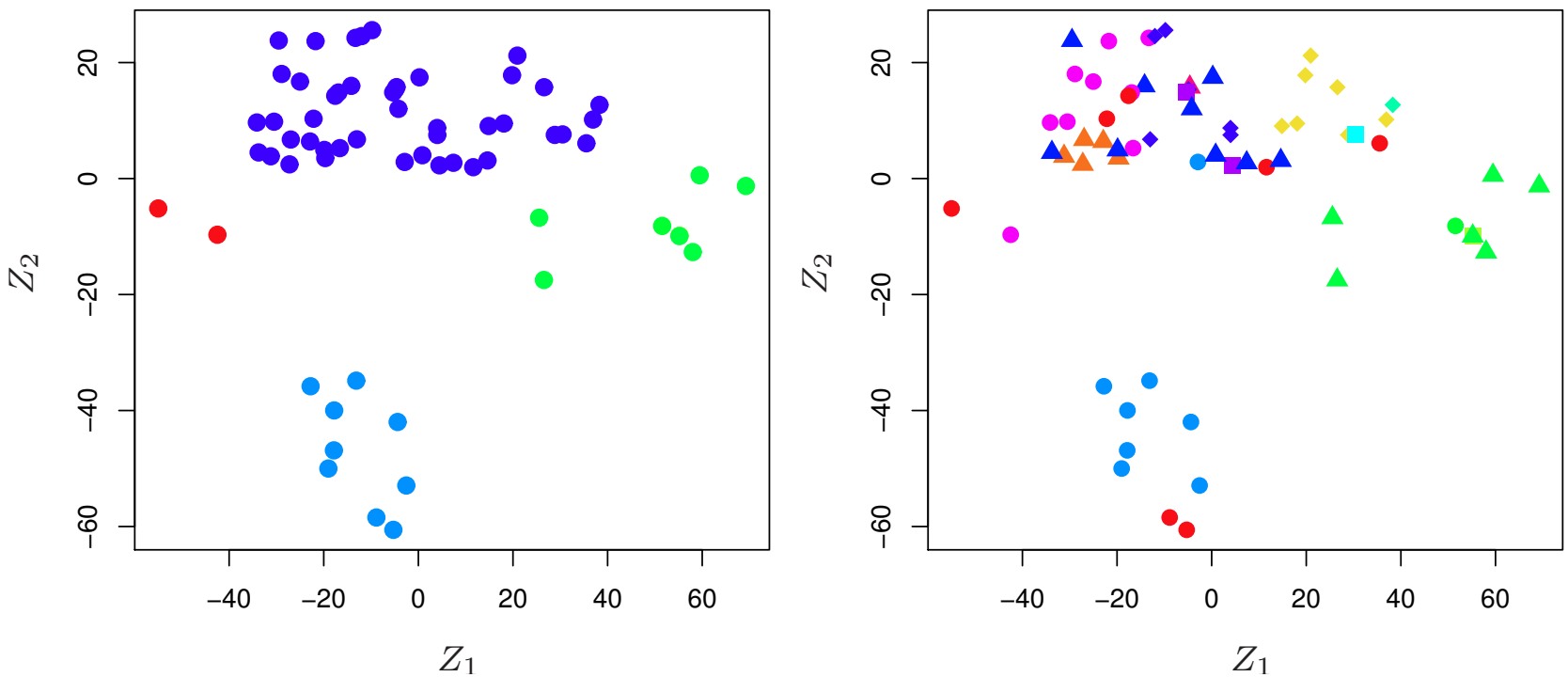
- Instances *within* the group are similar to each other;
- Instances *in different groups* are different from each other.
- Used to discover *structure* on the basis of a the data set.

Requires a definition of what it means for two instances to be considered *different* or *similar* to each other.

Therefore, it requires domain-specific expertise of the data being studied.

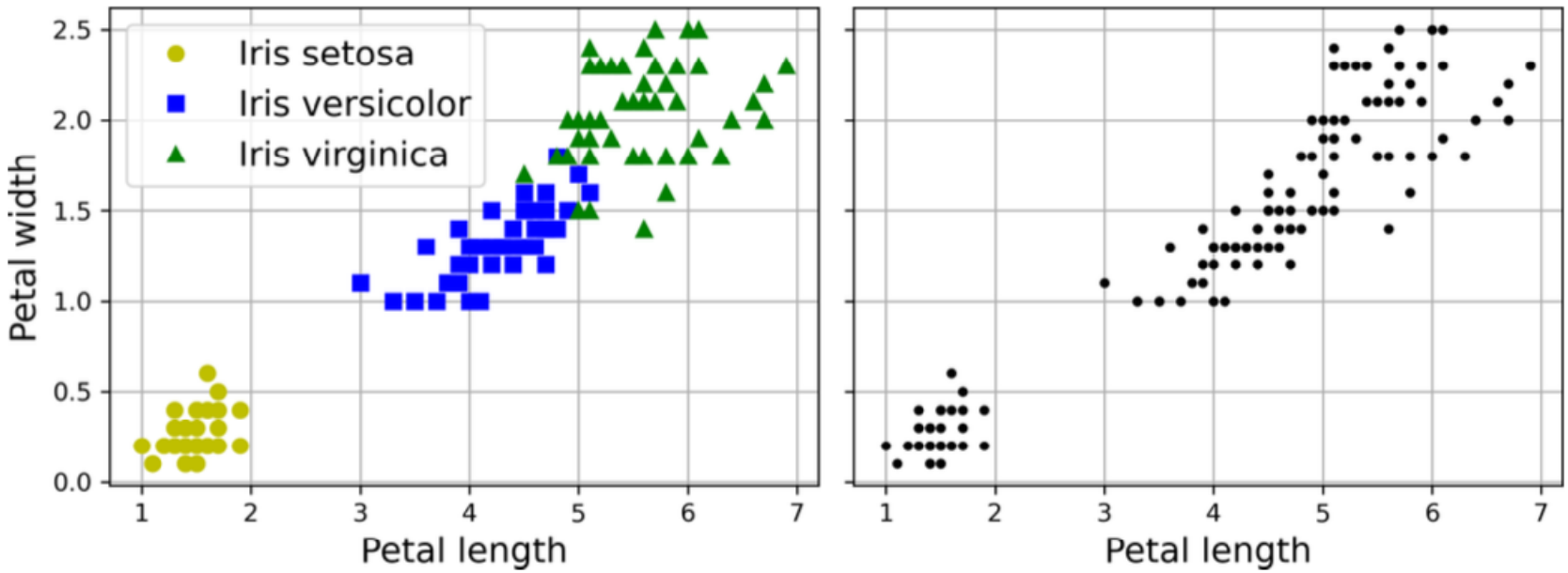
Clustering - Example

Considering the **NCI60** data set: $6,830$ gene expression measurements for the 64 cancer cell lines. The aim is to find whether there are groups among the cell lines based on their gene expression measurements. Cancer type is also recorded.



Each point corresponds to one of the 64 cell lines projected in a low-dimensional space, Z_1 and Z_2. Left: four colours for what seems to be four groups of cell lines. Right: Same as left panel, but colours now represent each of the 14 different types of cancer.

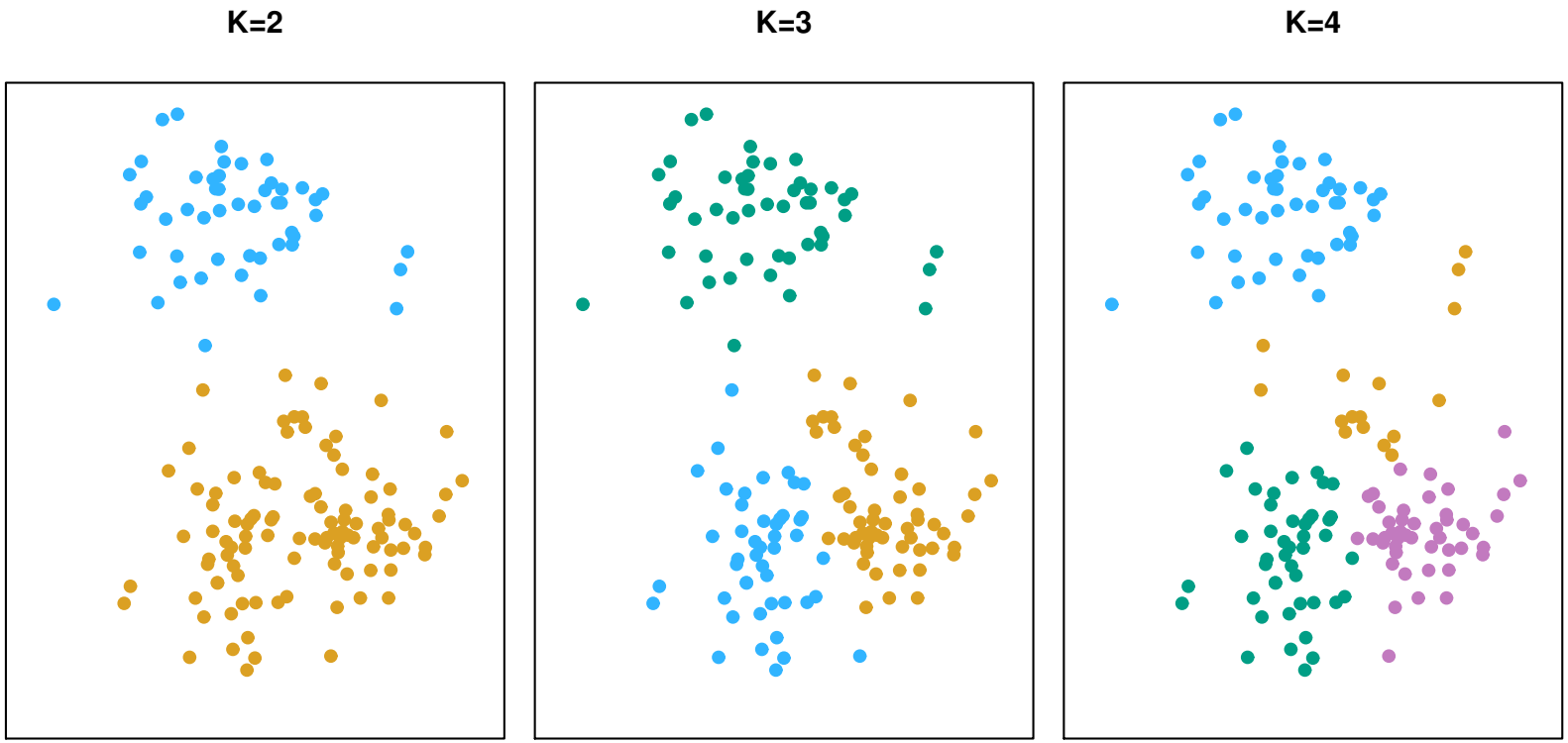
Clustering - Example



Left: classification: each instance's species (i.e., its class) is represented with a different marker. Right: clustering: the same data set, but with no labels.

K-means

K-means clustering partitions the data set into K distinct, non-overlapping clusters. Ideally, we want to minimise the variation within each K cluster and select an optimal value of K.



Simulated data set with 150 observations in two-dimensional space. From left to right: results of applying K-means clustering with values of K (the number of clusters) equal to 2, 3 and 4, respectively. The colour of each instance indicates the cluster to which it was assigned using the K-means clustering algorithm.

K-means

The sets $C_1, ..., C_K$ contain the indices of the instances belonging to each K cluster. These sets have two properties:

- $C_1 \cup C_2 \cup ... \cup C_K = \{1, ..., m\}$. Each instance belongs to at least one of the K clusters. E.g., $i \in C_k$ if the ith instance belongs to the kth cluster.
- $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$. Non-overlapping clusters: no instance belongs to more than one cluster.

K-means clustering aims to find a *good* clustering: one for which the `within-cluster variation` (the amount by which the instances within a cluster differ from each other) is as small as possible.

$$\text{minimise}_{C_1,...,C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

K-means

We need a definition for `within-cluster variation`. A popular choice is the *squared Euclidean distance*:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{n} \left(x_{ij} - x_{i'j}\right)^2$$

where $|C_k|$ is the total number of instances in the kth cluster.

Therefore, we want to solve the optimisation problem:

$$\text{minimise}_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{n} \left(x_{ij} - x_{i'j}\right)^2 \right\}$$

The K-means Algorithm

There are about K^m ways to partition m instances into K clusters, which is a huge number, even for small values of K and m.

The K-means algorithm provides a way to achieve a local optimum to the K-means optimisation problem:

1. Randomly assign a number, from 1 to K, to each instance to serve as initial cluster assignments for the instances.
2. Iterate until the cluster assignments stop changing:
   a. For each of the K clusters, compute the cluster *centroid*. The kth cluster centroid is a n-dimensional vector with the feature means for the instances in the kth cluster (n is the number of features).
   b. Assign each instance to the cluster whose centroid is closest. Usually *closest* is defined using Euclidean distance.

The K-means Algorithm

The algorithm is guaranteed to decrease the value of the objective of the optimisation problem at each step.

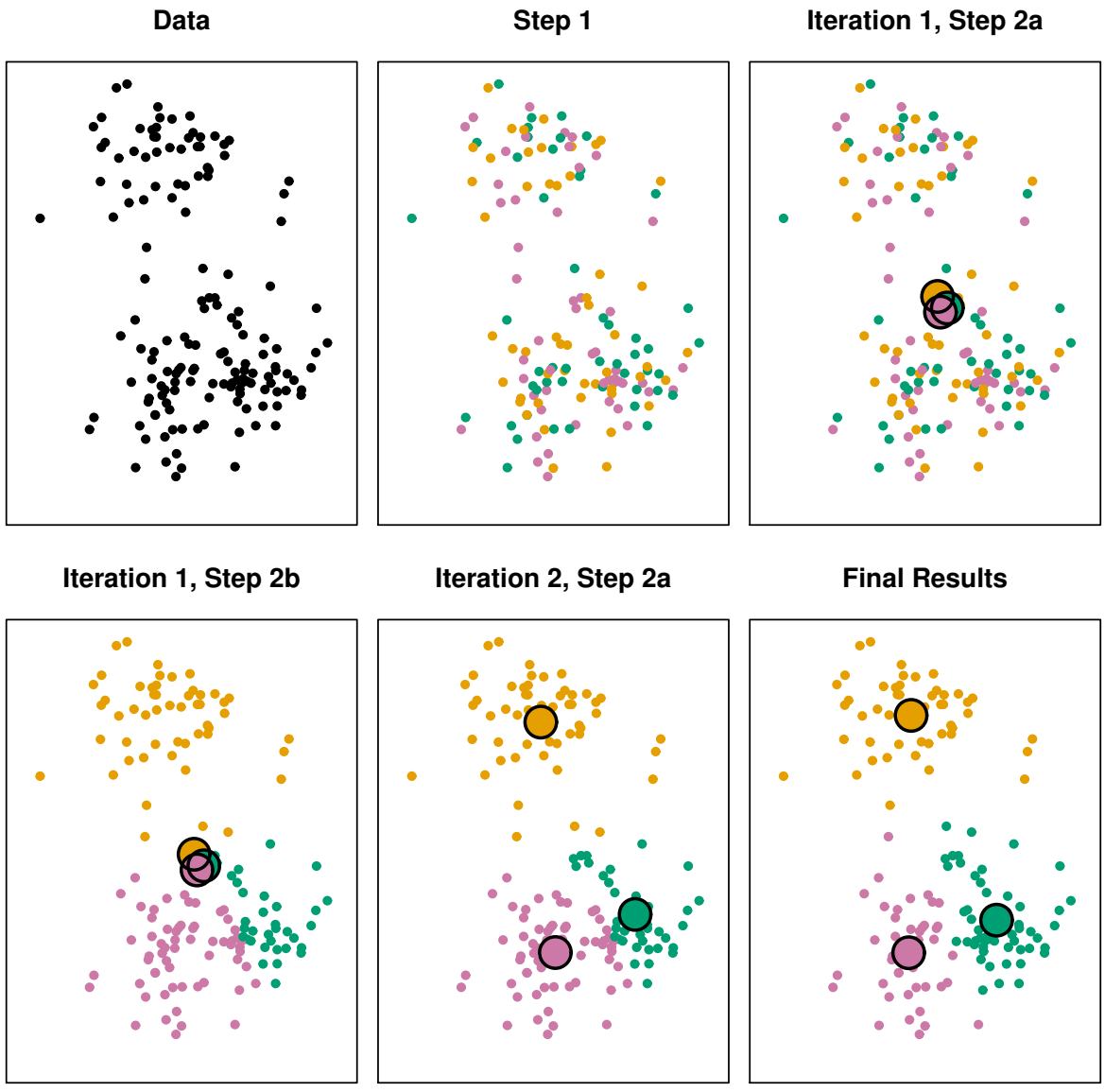
As the algorithm runs, the obtained clustering will continually improve until the result stops changing.

When the results stop changing, a *local optimum* has been reached.

The algorithm finds a local rather than a global optimum; therefore the results may differ each time we run it as they will depend on the initial (random) cluster assignment of each instance in Step 1 of the algorithm.

It is common to run the algorithm multiple times and select the best solution (smallest within-cluster variation).

The K-means Algorithm - Example

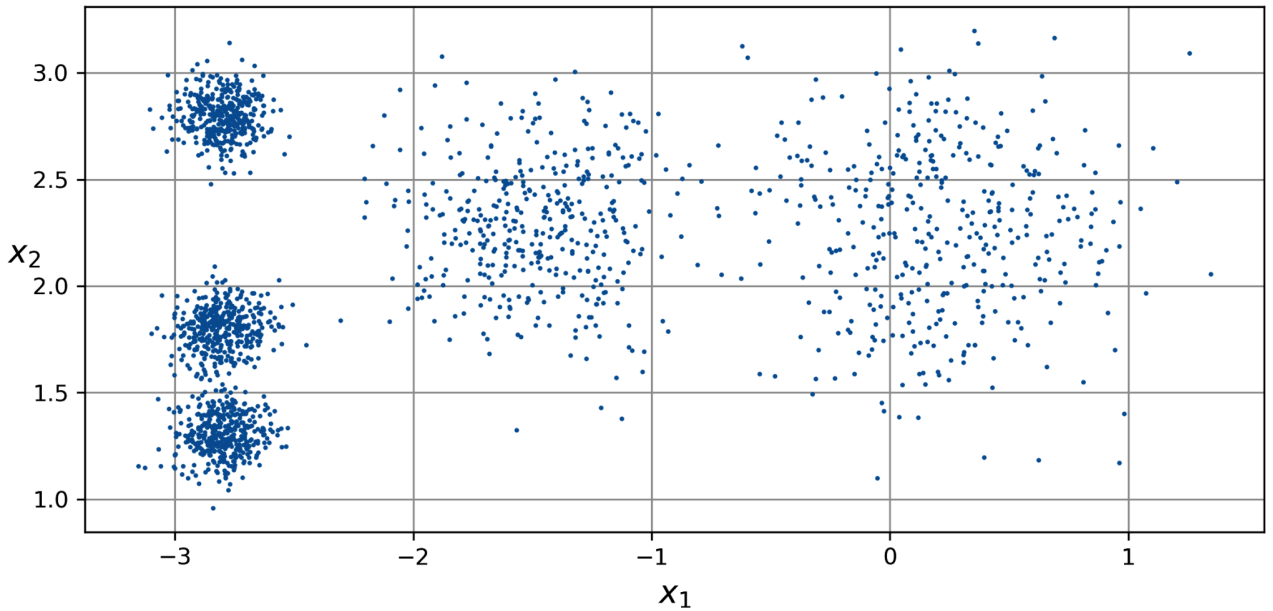


An example for a data set with 150 points and with $K = 3$.

The K-means Algorithm - Several Runs

The outcome of the K-means clustering with six different random assignments.

The K-means Clustering in Scikit-Learn



An unlabelled dataset composed of five blobs of instances

```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

X, y = make_blobs([...]) # make the blobs: y contains the cluster IDs, but we
                         # will not use them
k=5
kmeans = KMeans(n_clusters=k, random_state=42)
y_pred = kmeans.fit_predict(X)
```

The K-means Clustering in Scikit-Learn

Each instance is assigned to one of the five clusters (the instance's label is the index of the cluster to which this instance belongs according to the algorithm).

We can inspect the five centroids that the algorithm found:

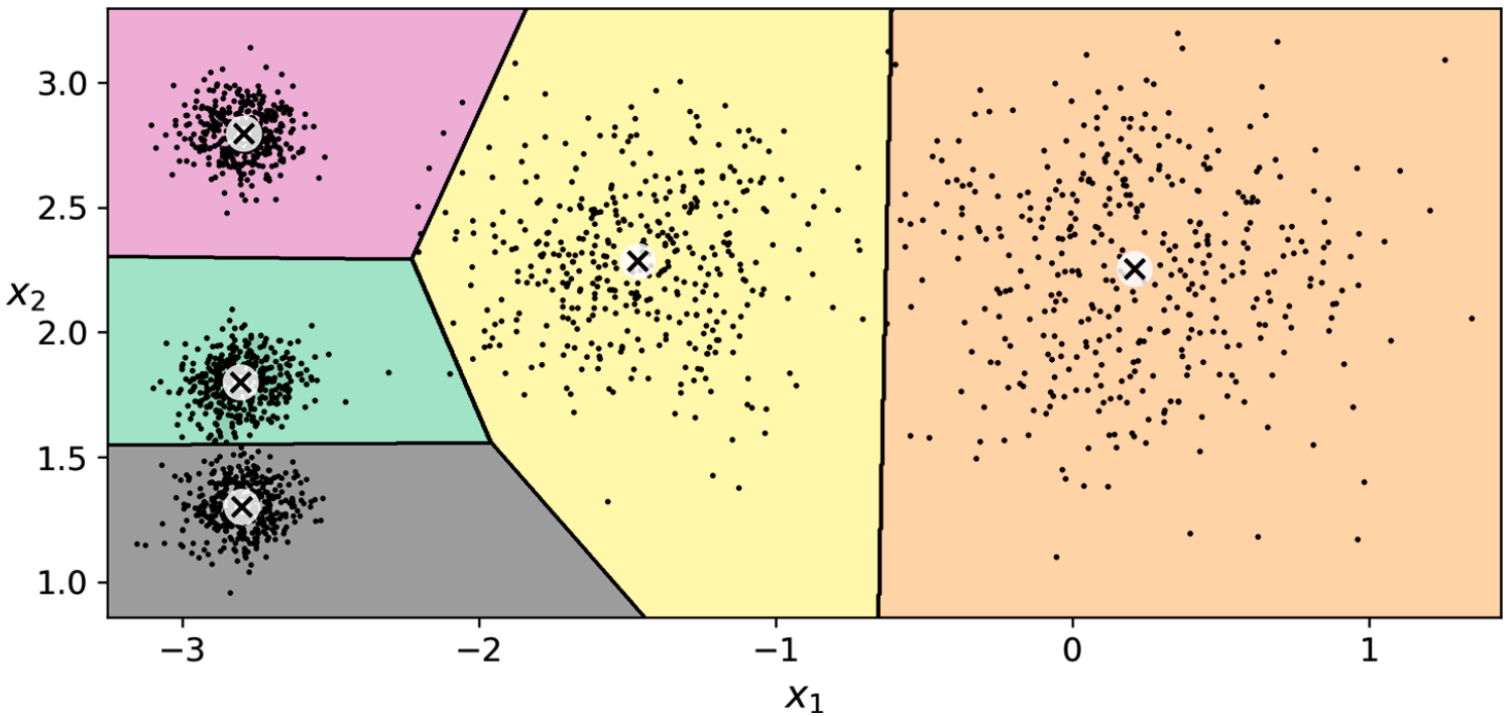
```
>>> kmeans.cluster_centers_
array([[-2.80389616, 1.80117999],
       [ 0.20876306, 2.25551336],
       [-2.79290307, 2.79641063],
       [-1.46679593, 2.28585348],
       [-2.80037642,  1.30082566]])
```

And assign new instances to the cluster whose centroid is closest:

```
>>> import numpy as np
>>> X_new = np.array([[0, 2], [3, 2], [-3, 3], [-3, 2.5]])
>>> kmeans.predict(X_new)
array([1, 1, 2, 2], dtype=int32)
```

The K-means Clustering in Scikit-Learn

The cluster's decision boundaries:

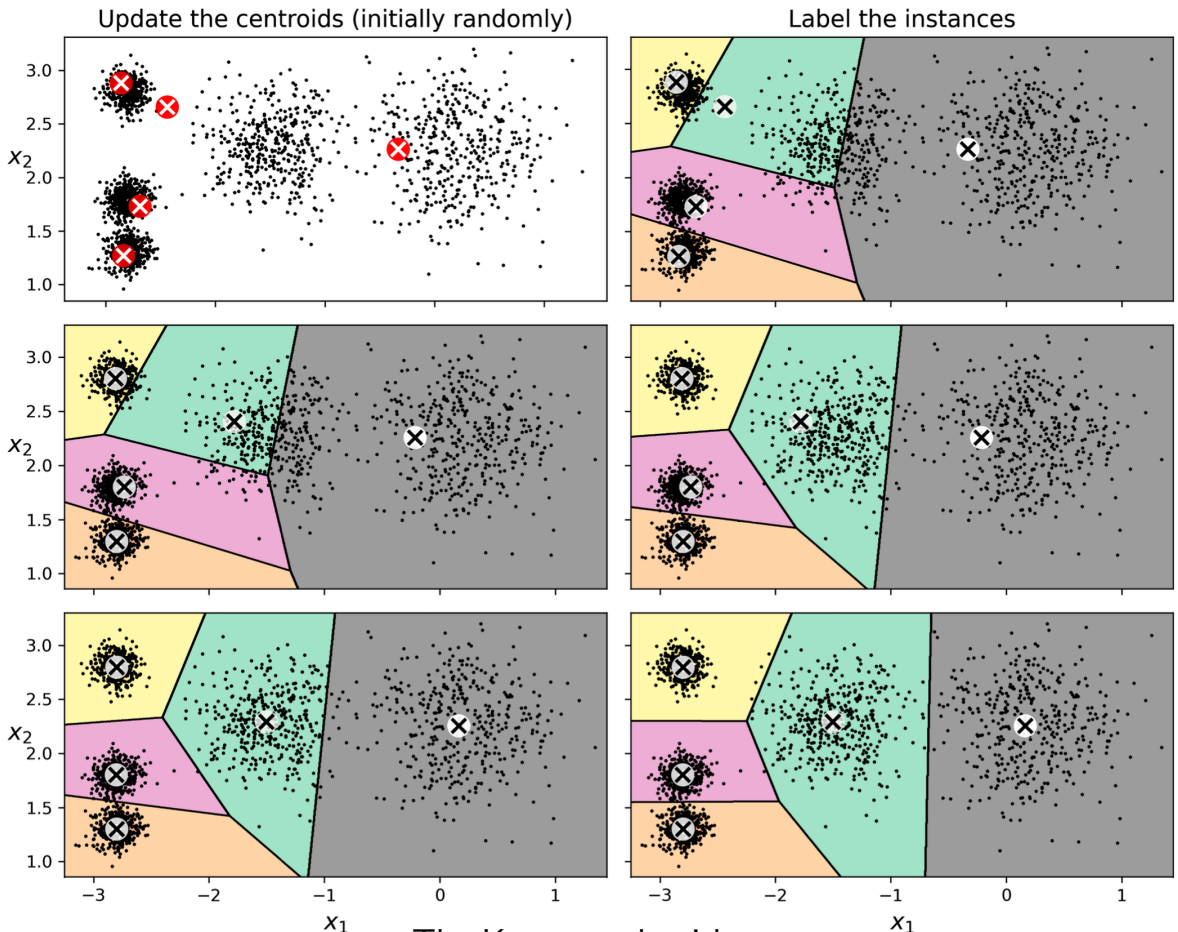


K-means decision boundaries (Voronoi tessellation)

As assigning an instance to a cluster is based only on the distance to the centroid, the algorithm may not perform very well when the blobs have very different diameters.

The K-means Clustering in Scikit-Learn

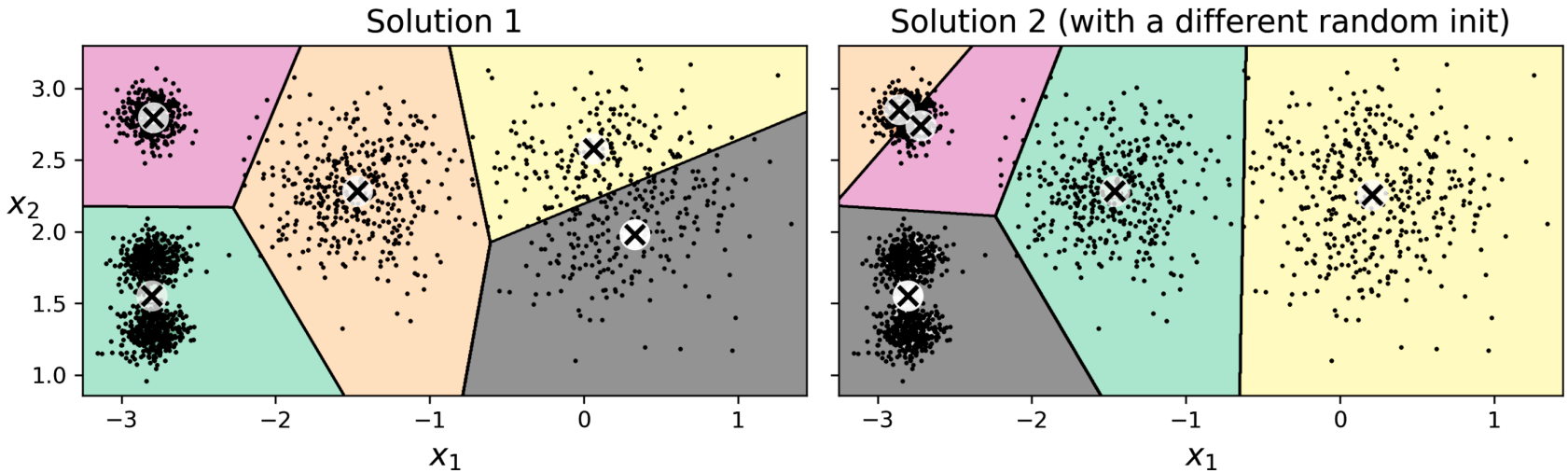
An alternative in the (random) initialisation step is to place the centroids randomly (e.g., by picking K instances at random from the data set and using their locations as centroids).



The K-means algorithm.

The K-means Clustering in Scikit-Learn

Although the algorithm is guaranteed to converge (the mean squared distance between the instances and their closest centroids can only go down at each step), the result depends on the centroid initialisation.



Suboptimal solutions due to unlucky centroid initialisations.

There are a few ways to improve the centroid initialisation.

Minimising the risk of not ideal initialisations

In Scikit-Learn, the number of random initialisations is controlled by the n_init hyperparameter (default value is 10). The K-means clustering algorithm is executed 10 times (i.e., with 10 different initialisations), and Scikit-Learn keeps the best solution.[1]

If better centroids are available (e.g. by using the result of another clustering algorithm), we can set the init hyperparameter and specify the list of centroids (then set n_init to 1):

```
good_init = np.array([[-3, 3], [-3, 2], [-3, 1], [-1, 2], [0, 2]])
kmeans = KMeans(n_clusters=5, init=good_init, n_init=1, random_state=42)
kmeans.fit(X)
```

[1] It uses the model's *inertia* as the performance metric.

Other K-means Variants

***K*-means++:** Proposed in 2006 by David Arthur and Sergei Vassilvitskii[2]. In the initialisation step, the selected centroids tend to be distant from one another (default initialisation in the Kmeans class).

Accelerated *K*-means: Proposed in 2023 by Charles Elkan[3]. It avoids unnecessary distance calculations by exploiting the triangle inequality and lower and upper bounds of the distance between instances and centroids to accelerate the *K*-means algorithm. To try, set algorithm = "elkan" in the Kmeans class.

Elkan's algorithm does not always speed up the algorithm, it depends on the data set.

[2] David Arthur and Sergei Vassilvitskii, "k-Means++: The Advantages of Careful Seeding", Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (2007): 1027–1035.

[3] Charles Elkan, "Using the Triangle Inequality to Accelerate k-Means", Proceedings of the 20th International Conference on Machine Learning (2003): 147–153.

Other K-means Variants

Mini-batch K-means: Proposed in 2010 by David Sculley[4]. It uses mini-batches instead of the full data set at each iteration, meaning the centroids will tend to move slightly at each iteration.

This speeds up the algorithm, and it makes it possible to cluster huge datasets that do not fit in memory (at the cost of a slightly worse performance).

```
from sklearn.cluster import MiniBatchKMeans

minibatch_kmeans = MiniBatchKMeans(n_clusters=5, random_state=42)
minibatch_kmeans.fit(X)
```

[4] David Sculley, "Web-Scale K-Means Clustering", Proceedings of the 19th International Conference on World Wide Web (2010): 1177–1178.

How to Select the Value of K?

The problem of selecting K is far from simple.

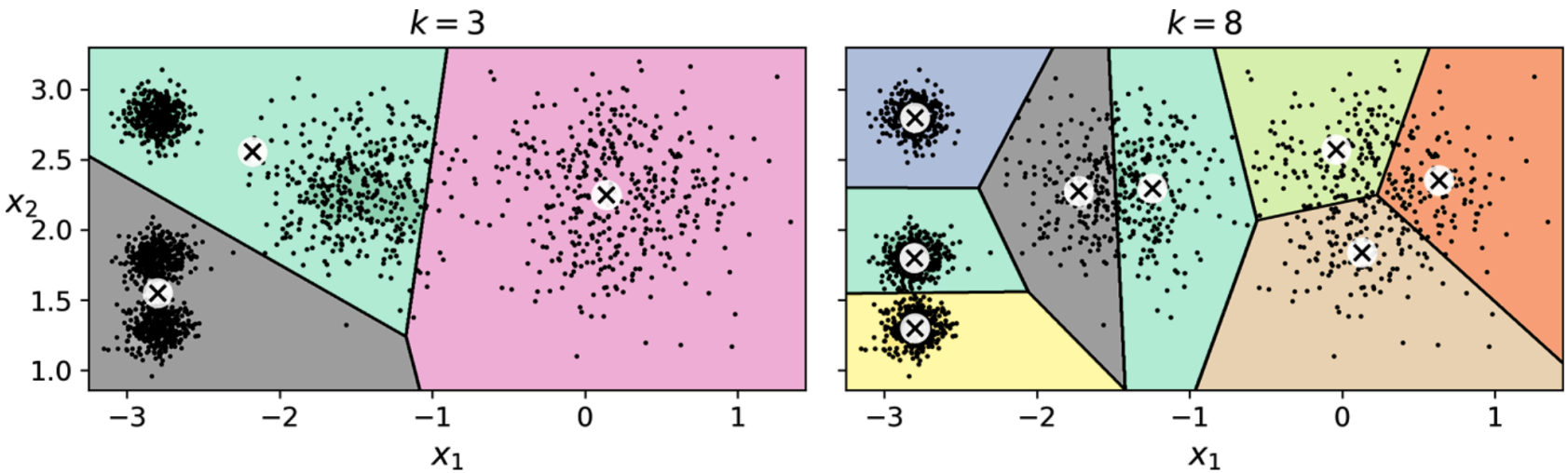
The obtained clusters should represent true subgroups in the data (and not simply a result expected by chance).

If we are given a new set of instances, then would these instances also display the same structure as the one obtained by the clustering algorithm (that is, the same set of clusters)?

Several techniques seek to assess or validate the evidence of the obtained clusters, but there has been no consensus on a single approach.

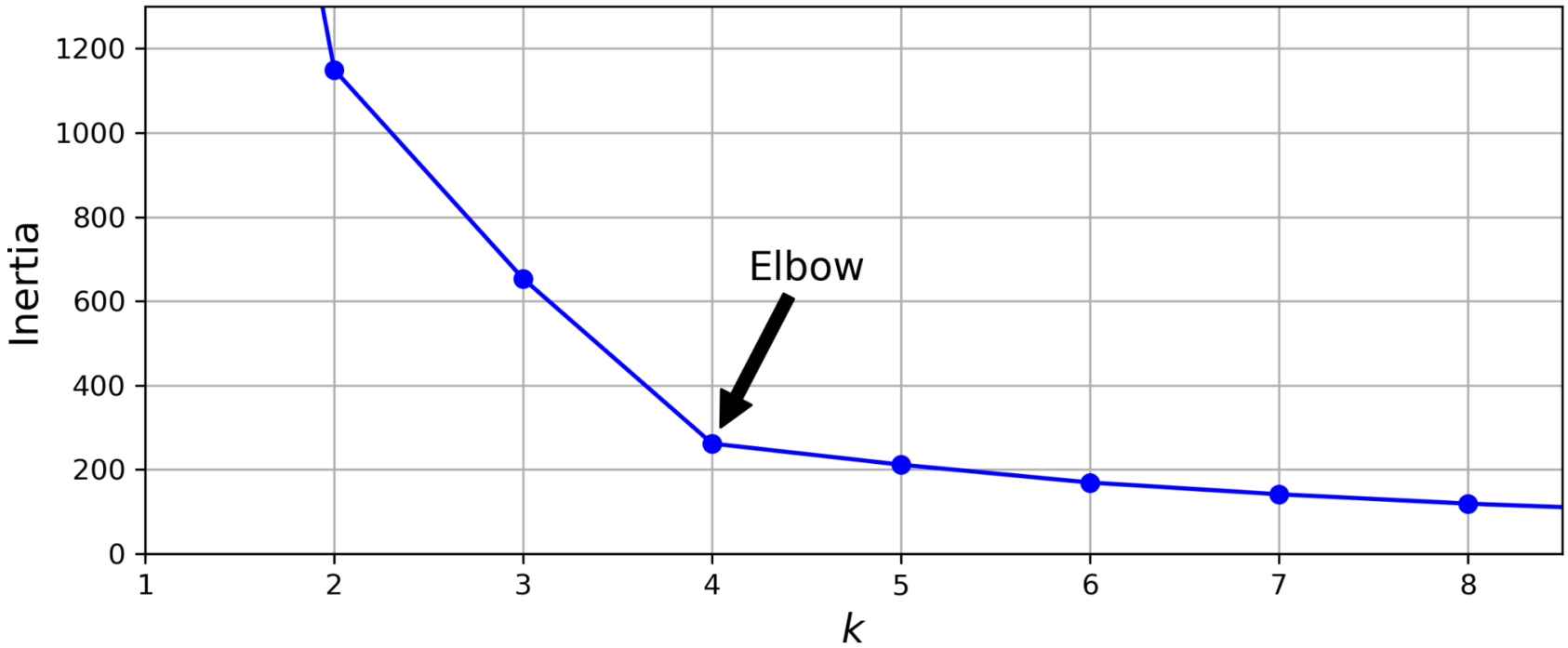
How to Select the Value of K?

Solutions for the previous blobs example for $K = 3$ and $K = 8$.



The impact on the clustering for different values of K, the number of clusters.

How to Select the Value of K?



Plotting the inertia as a function of the number of clusters

The `inertia` (the sum of the squared distances between the instances and their closest centroids) decreases as the number of clusters increases. That is, the more clusters there are, the closer each instance will be to its closest centroid.

How to Select the Value of K?

An alternative approach is the *silhouette score*, which is the mean *silhouette coefficient* over all instances.

The *silhouette coefficient* of an instance i is defined as

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

a is the mean distance to other instances in the same cluster (i.e., the instance's mean *intra*-cluster distance), and

b is the mean distance to the instances of the next closest cluster (i.e., the instance's mean *nearest*-cluster distance, defined as the one that minimises b, excluding the instance's own cluster).

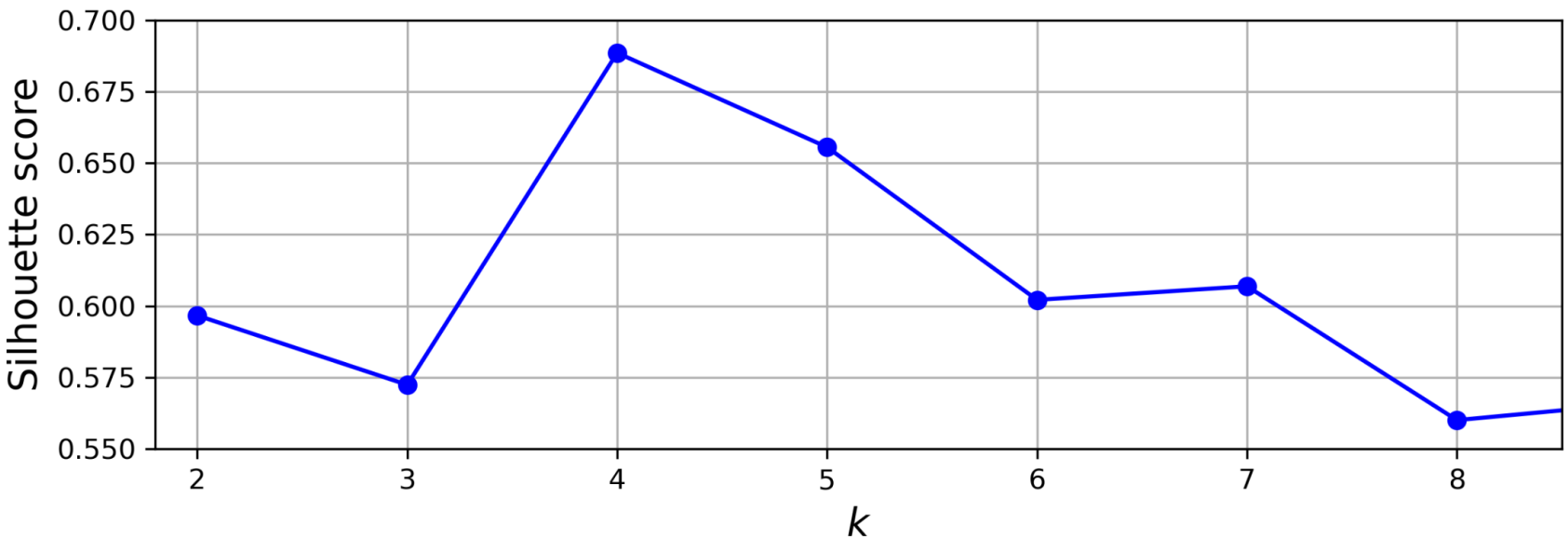
How to Select the Value of K?

The silhouette coefficient can vary between -1 and $+1$:

1: the instance is well inside its own cluster and probably far from other clusters.

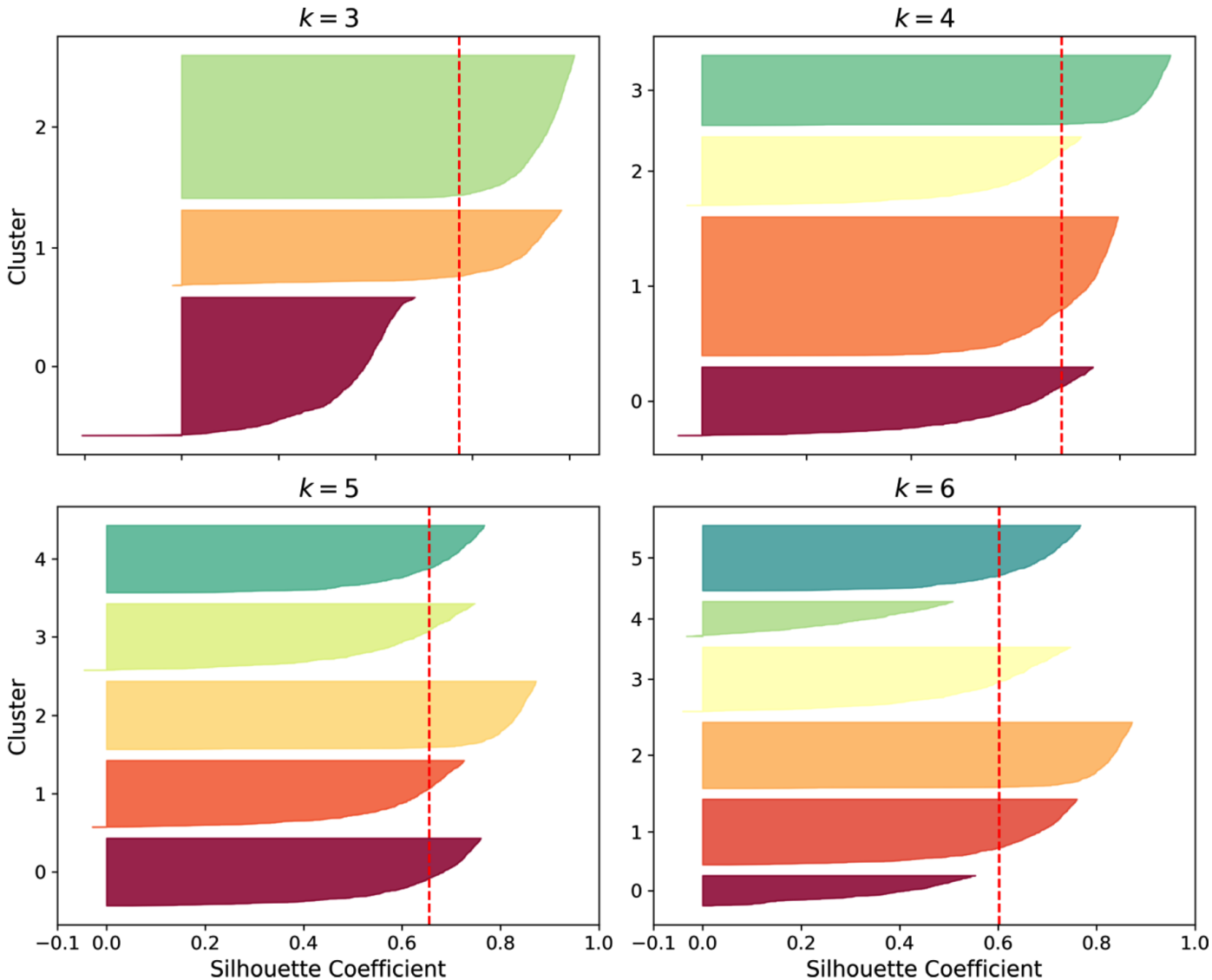
0: the instance may be close to a cluster boundary.

-1: the instance may have been placed in a wrong cluster.



Selecting the number of clusters using the silhouette score.

The Silhouette Diagram

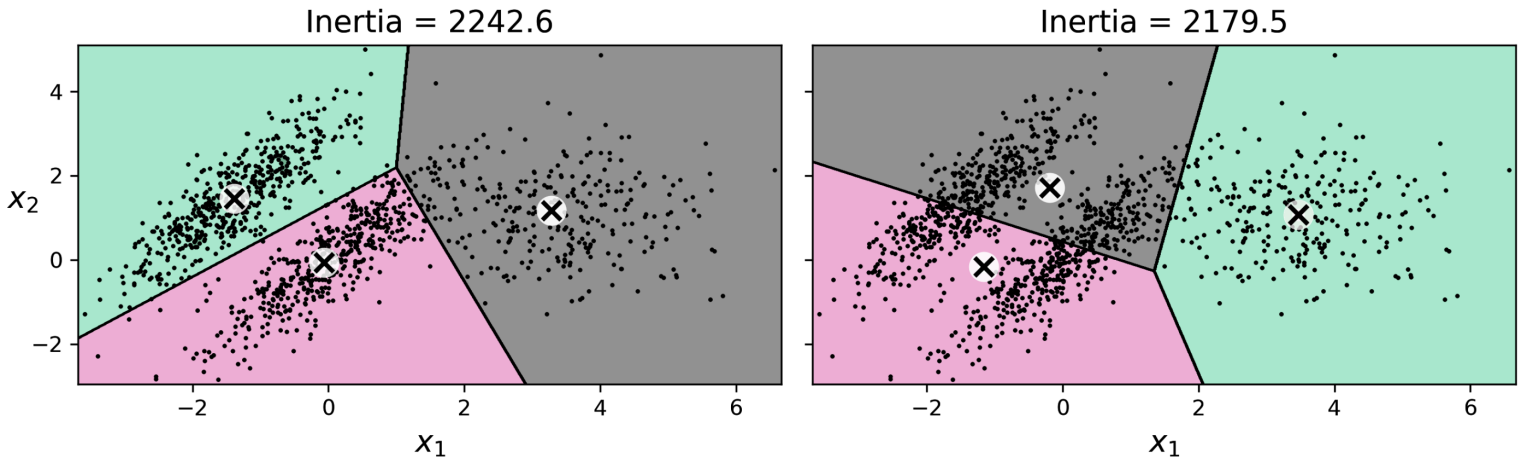


Analysing the silhouette diagrams for different number of clusters.

Limitations of K-means

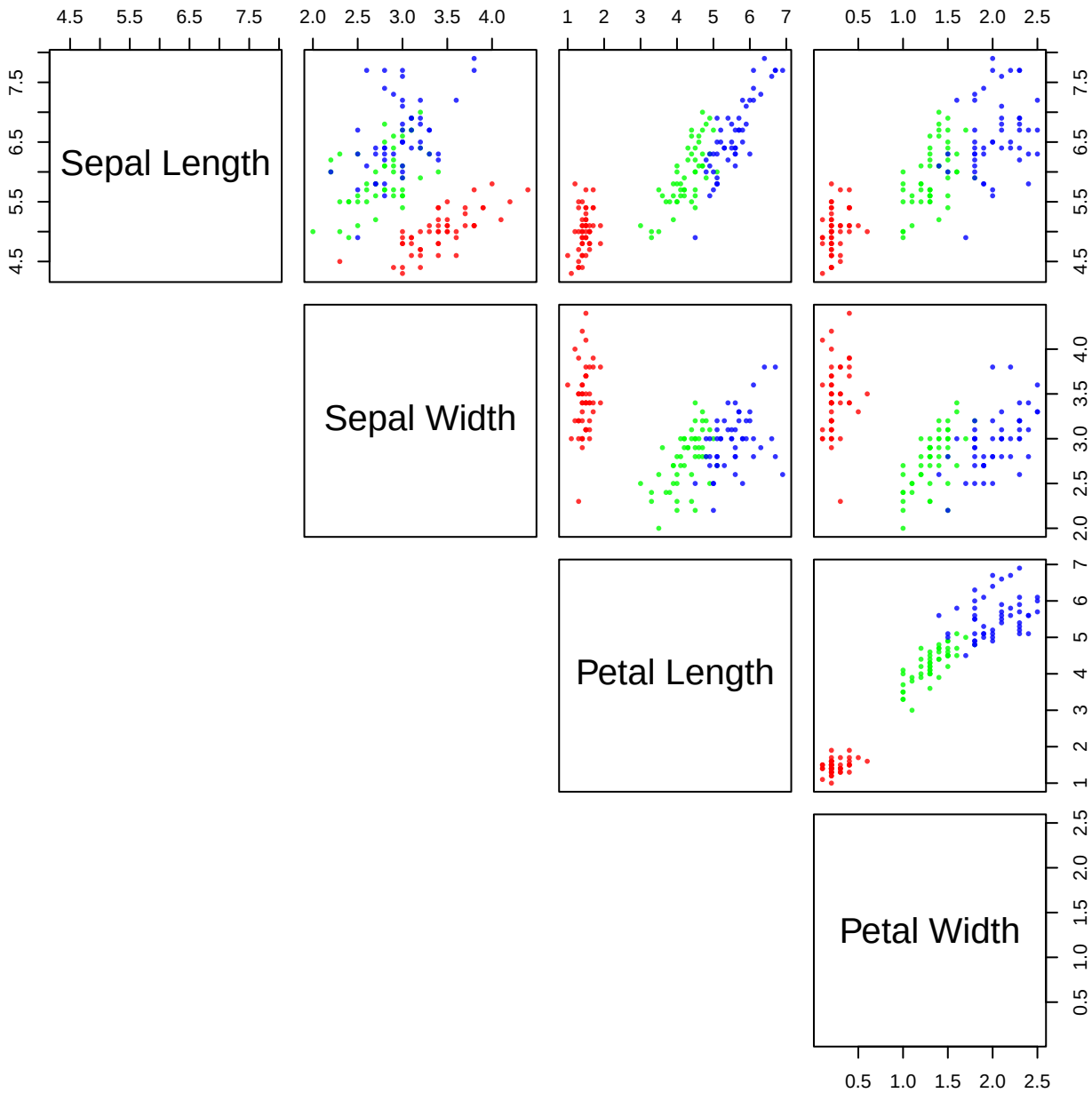
Some disadvantages of the K-means clustering algorithm include:

- Dependency on the initial (random) initialisation. We may need to run the algorithm several times to minimise the risk of suboptimal solutions.
- It requires a decision on the value of K, the number of clusters we expect in the data.
- It generally does not perform well for clusters with varying sizes, different densities, or nonspherical shapes. The data may require different clustering algorithms for better results.

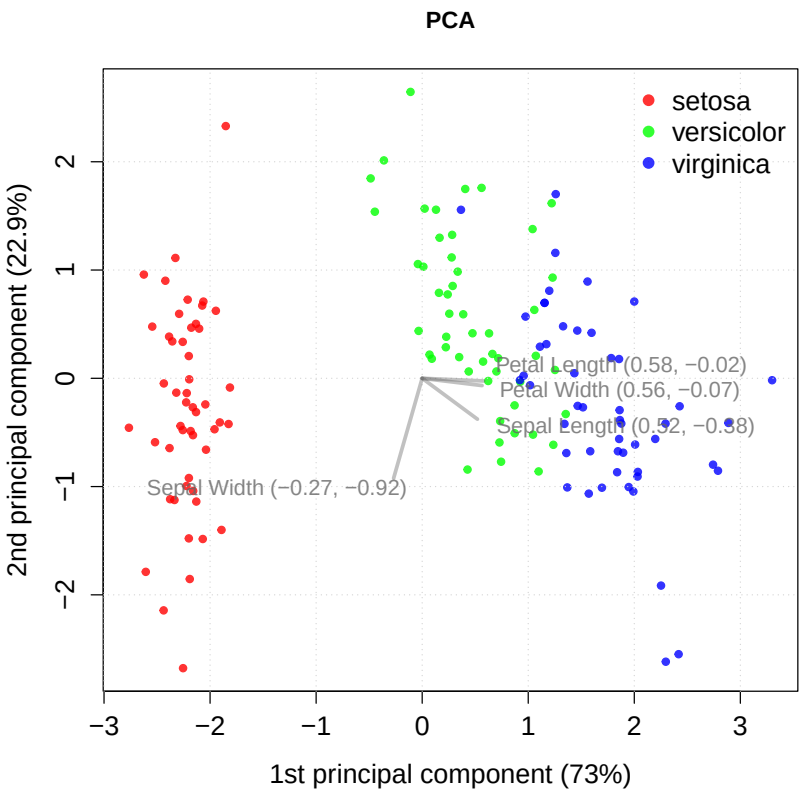


K-means on ellipsoidal blobs.

Discussion on the Iris Dataset



Discussion on the Iris Dataset



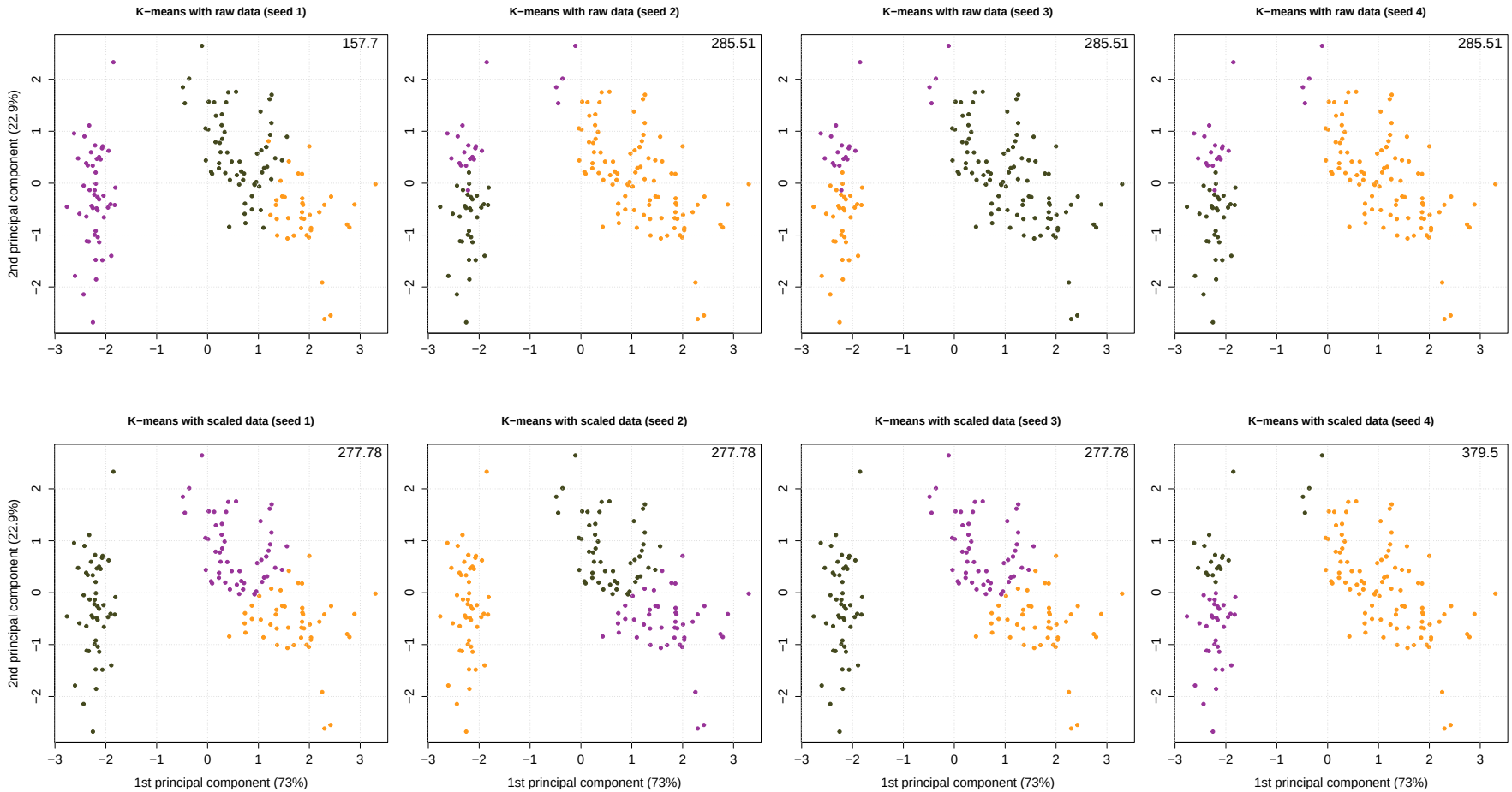
Mean of each variable per species

Variable	setosa	versicolor	virginica
Sepal Length	*5.006*	5.936	**6.588**
Sepal Width	**3.428**	*2.770*	2.974
Petal Length	*1.462*	4.260	**5.552**
Petal Width	*0.246*	1.326	**2.026**

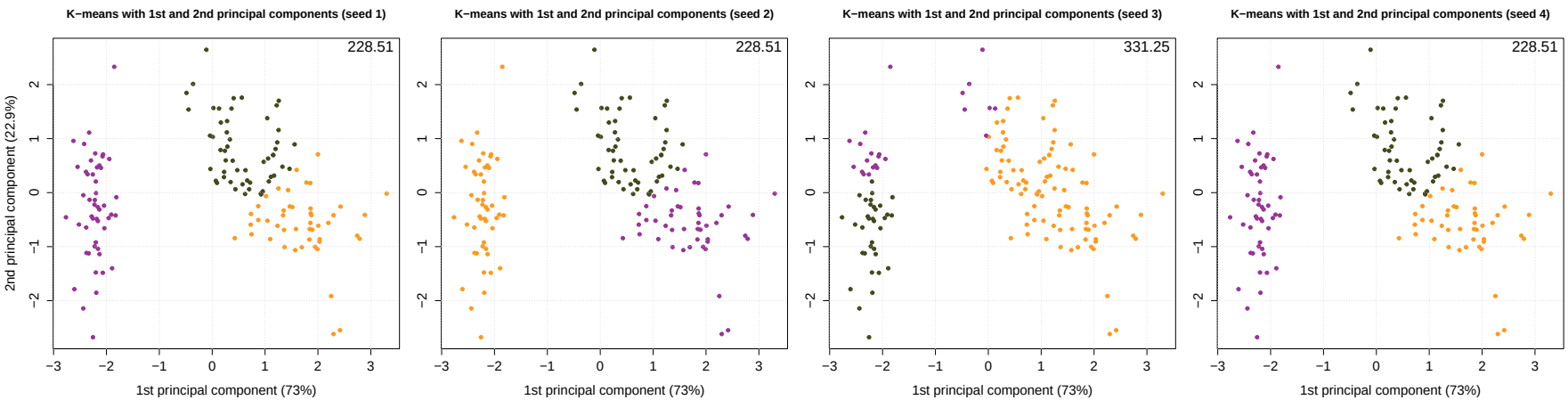
Mean of each scaled variable per species

Variable	setosa	versicolor	virginica
Sepal Length	*-1.011*	0.112	**0.899**
Sepal Width	**0.850**	*-0.659*	-0.191
Petal Length	*-1.301*	0.284	**1.016**
Petal Width	*-1.251*	0.166	**1.085**

Applying K-means on the Iris Dataset



Applying K-means on the 2 PCs of Iris Dataset



For next week

Work through Assignment 3 and attend a supervised lab. The Unit Coordinator or a casual Teaching Assistant will be there.

Read Chapter 12 on Clustering Methods in "An Introduction to Statistical Learning".