

Clustering

Clustering

- Aggarwal, Charu C. Data mining: the textbook. Springer, 2015.
- Steele, B., Chandler, J., & Reddy, S. (2016). Algorithms for data science (pp. 1-430). Springer.
- Cao, L. (2018). Data science thinking: The next scientific, technological and economic revolution. Data Analytics. Cham: Springer.
- Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc
- Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. Journal of intelligent information systems, 17(2), 107-145.



Unsupervised Learning: Clustering

- **Introduction and overview**
- k-Means clustering
- Finding the right number of clusters
- Scale issues
- Other cluster algorithms

- ▶ Clustering is an unsupervised technique to divide a set of objects into clusters (subgroups) such that within-cluster similarities are high and between-cluster similarities are small.

- ▶ With cluster analysis (clustering), we try to find
 - ▼ Unknown pattern in our observations
 - ▼ Homogeneous subgroups (also known as clusters)
 - ▼ Similarities between observations

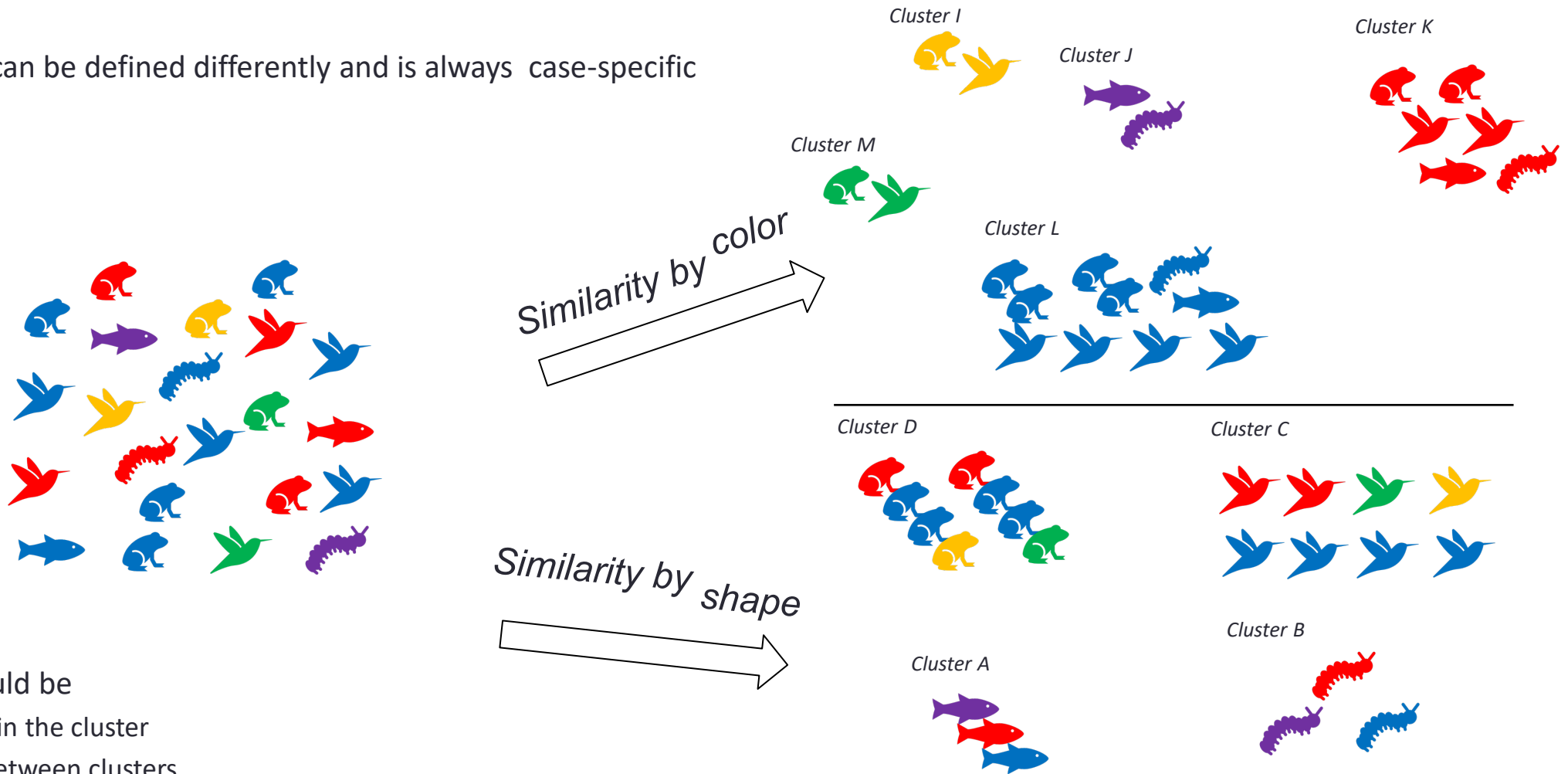
- ▶ Clustering is unsupervised learning:
 - ▼ No predefined classes
 - ▼ A form of learning by observation, rather than learning by examples

- ▶ How can we identify clusters?
 - ▼ High similarity within the cluster 
 - ▼ High dissimilarity between clusters 

- ▶ Typical applications for clustering
 - ▼ As a stand-alone tool to get insight into data, including visualization
 - ▼ As a preprocessing step for other algorithms (e.g., outlier detection, classification, data summarization)

Clustering // cluster analysis

- ▶ “Similarity” can be defined differently and is always case-specific



- ▶ Clusters should be
 - ▼ Similar within the cluster
 - ▼ Dissimilar between clusters

What is clustering? Central terms and context

- ▶ Cluster: a collection of observations / data objects
 - ▼ Observations are similar to one another within the same cluster and dissimilar to the objects in other clusters
 - ▼ A dataset is divided into several clusters as (disjoin) subsets
 - ▼ Alternative names: groups/subgroups

- ▶ Cluster analysis / clustering: the activity to...
 - ▼ find similarities between observations according to characteristics found in the data
 - ▼ grouping similar observations into clusters

- ▶ Cluster analysis is also known as unsupervised (machine) learning:
 - ▼ no predefined classes (ground truth data) existent
 - ▼ “unsupervised” means that object labels are not known in advance (no training data available)
 - ▼ leads to the discovery of previously unknown clusters
 - ▼ clustering is a form of learning by observation, rather than learning by examples
 - ▼ Unsupervised learning tries to find a hidden structure in “unlabeled data”

When and where to use clustering: Some examples and contexts

► Marketing

- ▼ E.g., Companies have access to a large number of measurements
 - median household income,
 - occupation,
 - distance from the nearest urban area,
 - technology affinity,
- ▼ Clustering helps marketers discover distinct groups in their customer bases and use this knowledge to develop targeted marketing programs

► Product recommendation

- ▼ You can cluster your customers based on their purchases and their activity on your website.
- ▼ This is useful to understand who your customers are and what they need, so you can adapt your products and marketing campaigns to each segment.
- ▼ Clustering can be useful in recommender systems to suggest content that other users in the same cluster enjoyed.

► Health

- ▼ E.g., Laboratories have various samples (= n observations) of cancer tissues with different features (z.B. tumor stage or grade, gene expression measurements) collected for each tissue sample
- ▼ Clustering could be used to find different clusters of cancer (types of cancer), which need different types of treatments

When and where to use clustering: Some examples and their use cases

- ▶ For anomaly detection (also called outlier detection)
 - ▼ Any instance that has a low affinity to all the clusters is likely to be an anomaly.
 - ▼ For example, if you have clustered the users of your website based on their behavior, you can detect users with unusual behavior, such as an unusual number of requests per second.
 - ▼ Anomaly detection is particularly useful in detecting defects in manufacturing, or for fraud detection.
- ▶ For semi-supervised learning
 - ▼ If you only have a few labels, you could perform clustering and propagate the labels to all the instances in the same cluster.
 - ▼ This technique can greatly increase the number of labels available for a subsequent supervised learning algorithm and thus improve its performance.
- ▶ For search engines
 - ▼ Some search engines let you search for images that are similar to a reference image.
 - ▼ To build such a system, you would first apply a clustering algorithm to all the images in your database; similar images would end up in the same cluster.
 - ▼ Then when a user provides a reference image, all you need to do is use the trained clustering model to find this image's cluster, and you can then simply return all the images from this cluster.
- ▶ For image segmentation
 - ▼ By clustering pixels according to their color, then replacing each pixel's color with the mean color of its cluster, it is possible to considerably reduce the number of different colors in the image.
 - ▼ Image segmentation is used in many object detection and tracking systems, as it makes it easier to detect the contour of each object.

Different clustering methods

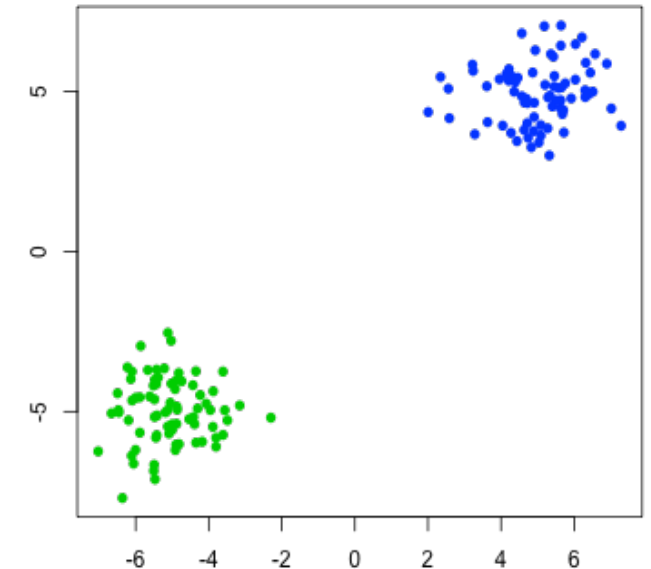
- ▶ Since clustering is popular in many fields, there exists a great number of clustering methods.
- ▶ Requirements for a “perfect” cluster methods
 - ▼ Ability to deal with large amount of data
 - ▼ Ability to deal with different types of attributes
 - ▼ Ability to discover clusters with arbitrary shape
 - ▼ Ability to deal with noisy data
 - ▼ Ability to deal with a large number of attributes
 - ▼ Ability to generate interpretable results
- No cluster method fulfills all requirements
- ▶ We focus on the best-known clustering algorithm: K-means clustering

Unsupervised Learning: Clustering

- Introduction and overview
- **k-Means clustering**
- Finding the right number of clusters
- Scale issues
- Other cluster algorithms

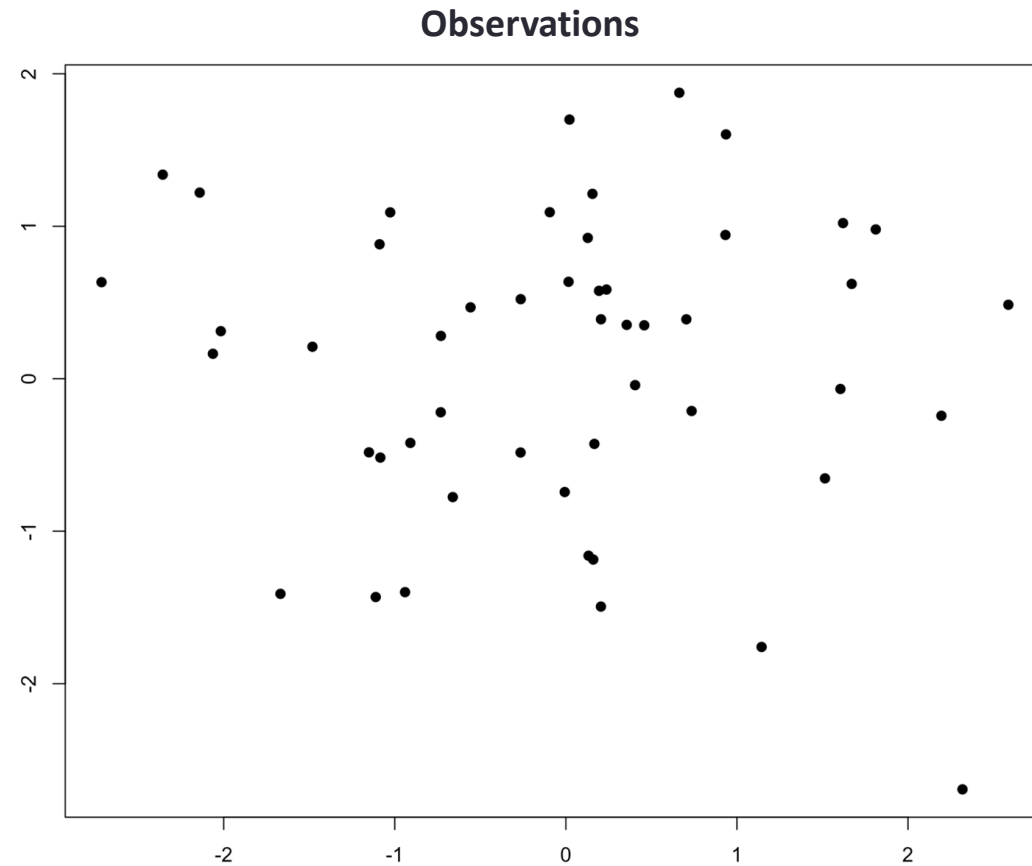
The k-means algorithm for clustering

- ▶ The k-means algorithm is perhaps the most often used clustering method
 - ▼ The foundation for many more sophisticated clustering techniques
 - ▼ The foundation of k-means will provide the knowledge to understand nearly any clustering algorithm in use today
- ▶ K-means divide a set of observations (e.g., customers) into a pre-defined number (k) of clusters
 - ▼ Clusters are represented by their cluster centers (centroid)
 - ▼ The number of clusters (k) must be specified at the start
 - ▼ The goal is to minimize the differences within each cluster and maximize the differences between clusters.
- ▶ K-means uses a heuristic process that finds locally optimal solutions
 - ▼ Starts with an initial guess for the cluster assignments then modifies the assignments slightly to see if the changes improve the homogeneity within the clusters
 - ▼ Common are the heuristics by Lloyd and MacQueen

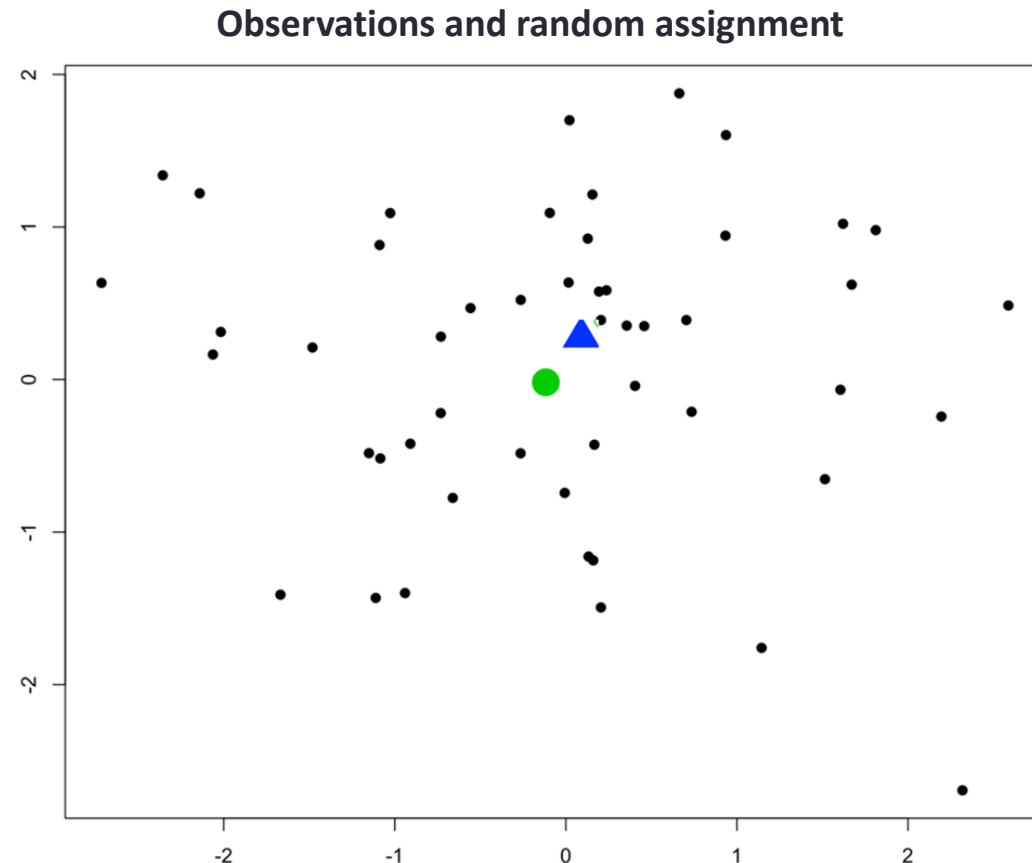


Example: k-Means clustering

► Goal: Partition data in k disjoint subsets



Example: k-Means clustering

► Goal: Partition data in k disjoint subsets

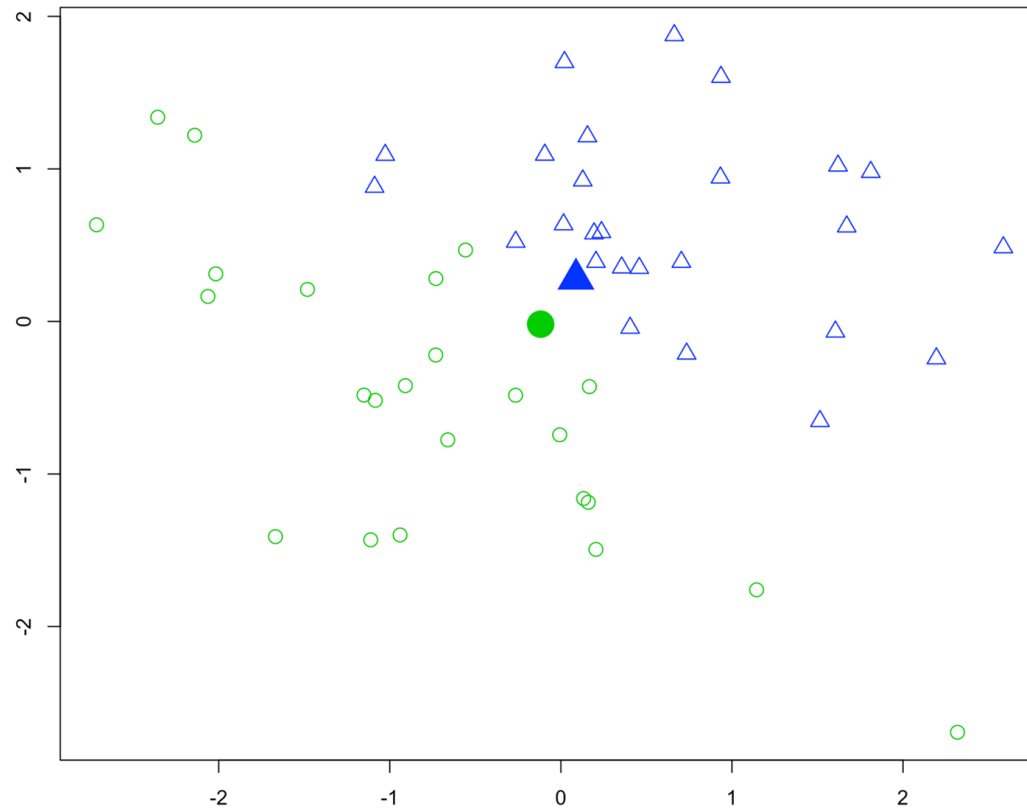
1. Randomly assign k centroids

► Explanation:

- ▼ The k-means algorithm begins by choosing k points in the feature space to serve as the cluster centroids. These centroids are the catalyst that spurs the remaining examples to fall into place. Because we hope to identify two clusters, $k = 2$ centroids are selected. These p centroids are indicated by the blue triangle and the green circle in the figure.
- ▼ There are several other ways to choose the initial cluster centers. One option is to choose random values occurring anywhere in the feature space (rather than only selecting among values observed in the data). Another option is to skip this step altogether; by randomly assigning each example to a cluster, the algorithm can jump ahead immediately to the update phase. Each of these approaches adds a particular bias to the final set of clusters, which you may be able to use to tailor your results.

Example: k-Means clustering

Assignment of observations to clusters

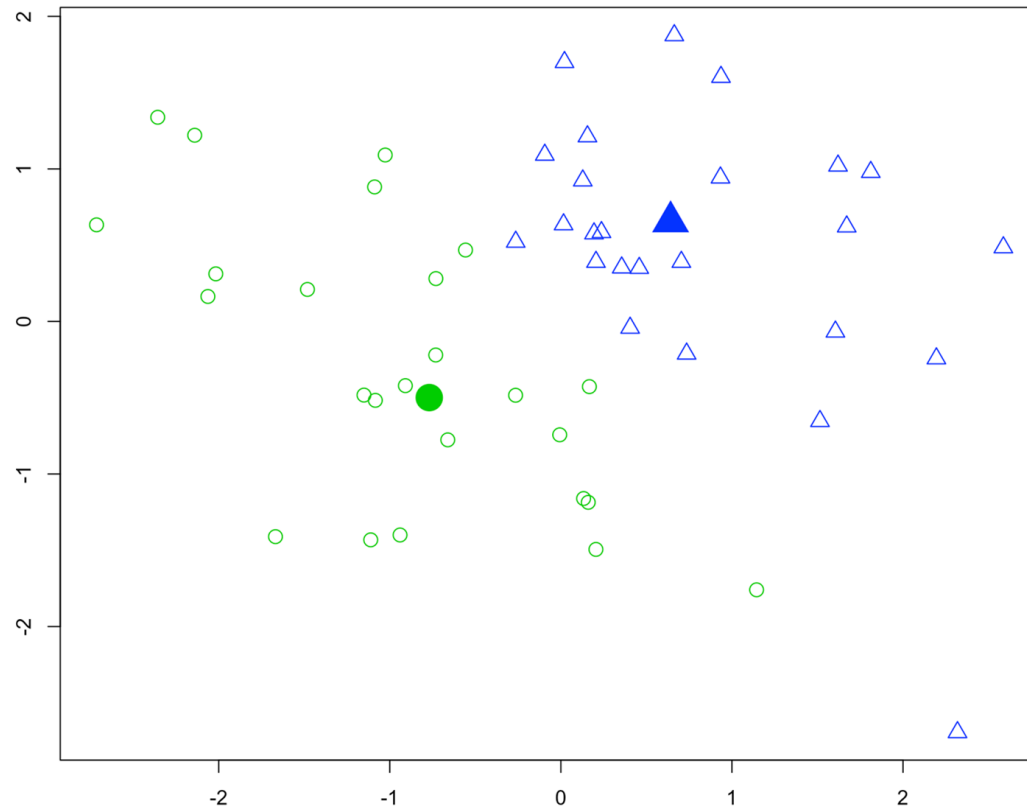


► Goal: Partition data in k disjoint subsets

1. Randomly assign k centroids
2. Assign each observation to the cluster whose centroid is closest
 - Mostly uses Euclidian distance function for numerical variables
 - Alternatively for numerical variables: Manhattan distance, Makowski distance, or Pearson Correlation distance
 - For categorical variables own dummy values can be created
 - For binary variables the Jaccard distance can be used
 - Different distant functions can lead to different cluster centers and different solutions.

Example: k-Means clustering

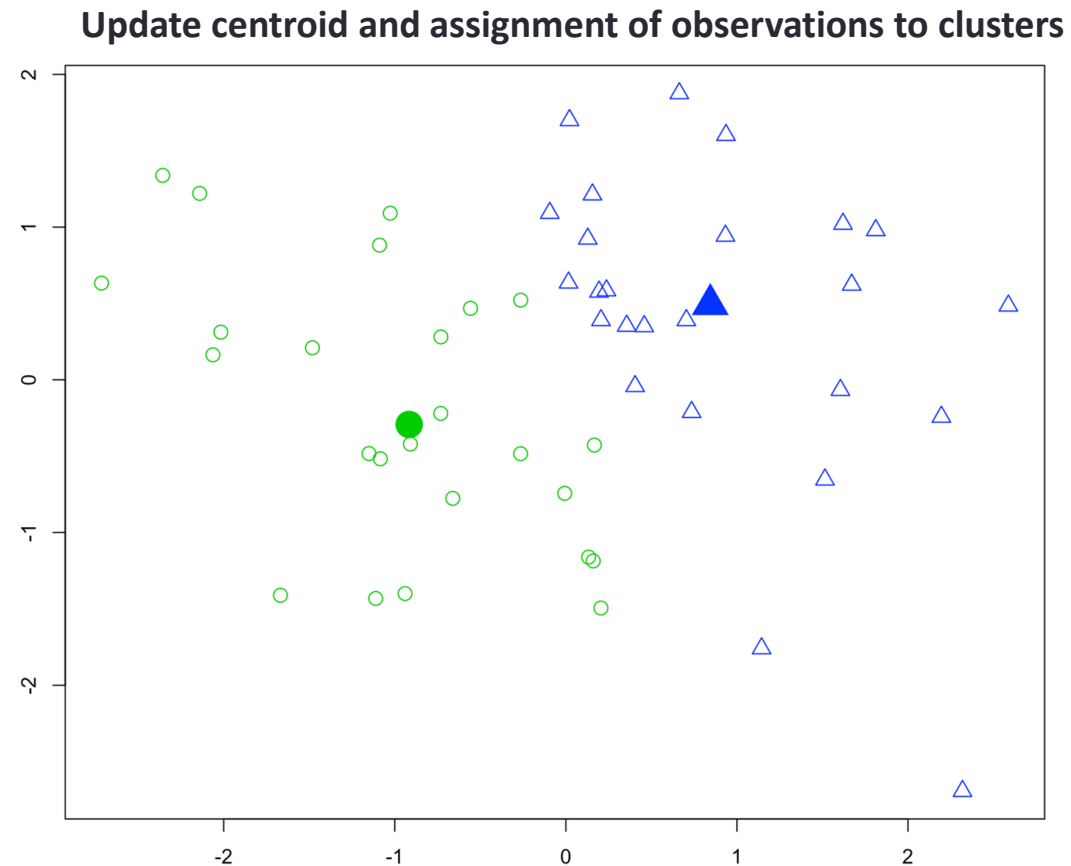
Update centroid and assignment of observations to clusters



► Goal: Partition data in k disjoint subsets

1. Randomly assign k centroids
2. Assign each observation to the cluster whose centroid is closest
3. Compute the new cluster centroids
 - The new centroid is calculated as the mean value of the observations currently assigned to that cluster.

Example: k-Means clustering

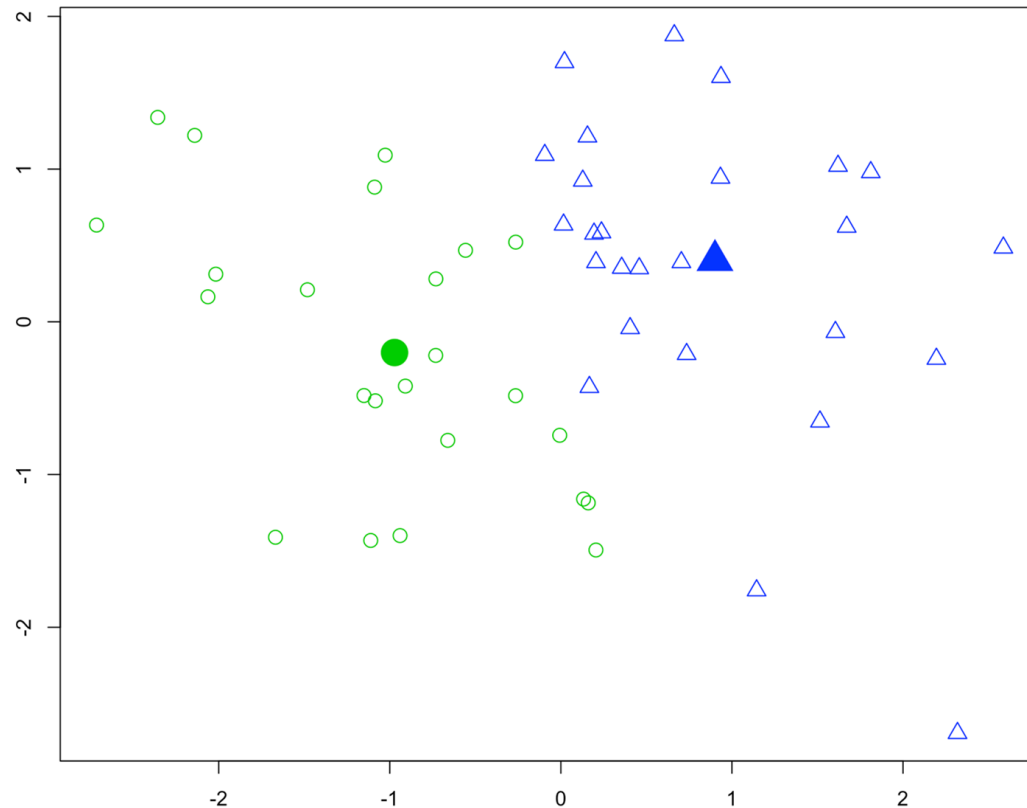


► Goal: Partition data in k disjoint subsets

1. Randomly assign k centroids
2. Assign each observation to the cluster whose centroid is closest
3. Compute the new cluster centroids
4. Repeat step 2 and 3 until the result no longer changes

Example: k-Means clustering

Update centroid and assignment of observations to clusters

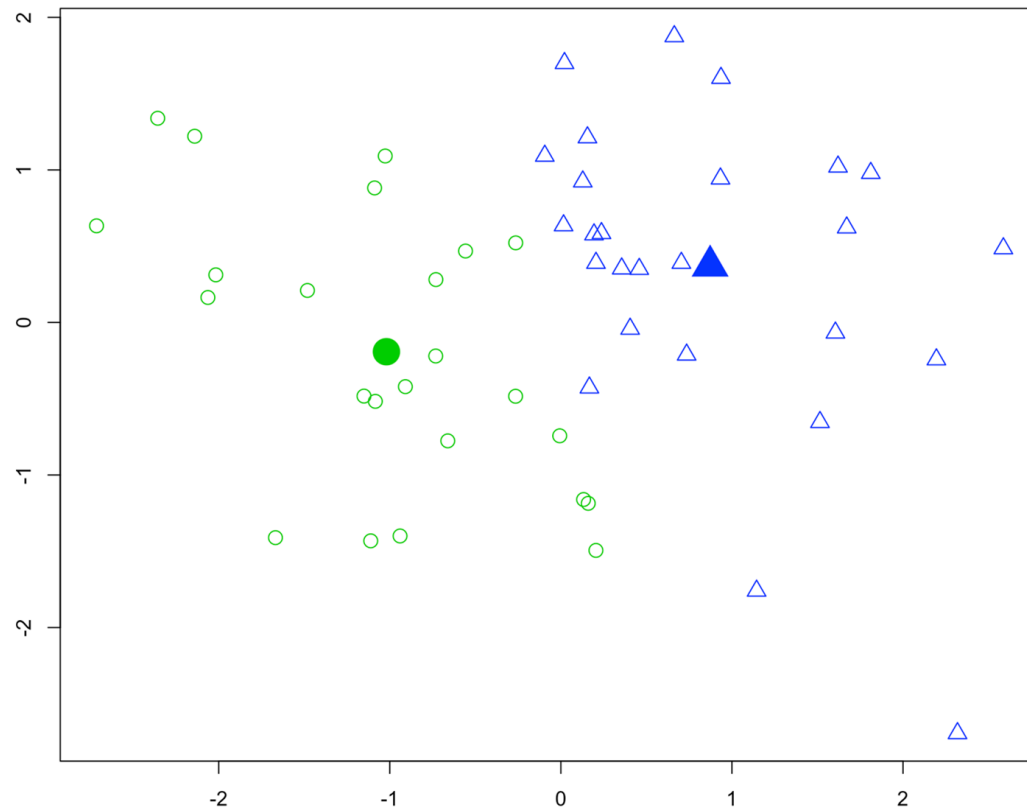


► Goal: Partition data in k disjoint subsets

1. Randomly assign k centroids
2. Assign each observation to the cluster whose centroid is closest
3. Compute the new cluster centroids
4. Repeat step 2 and 3 until the result no longer changes

Example: k-Means clustering

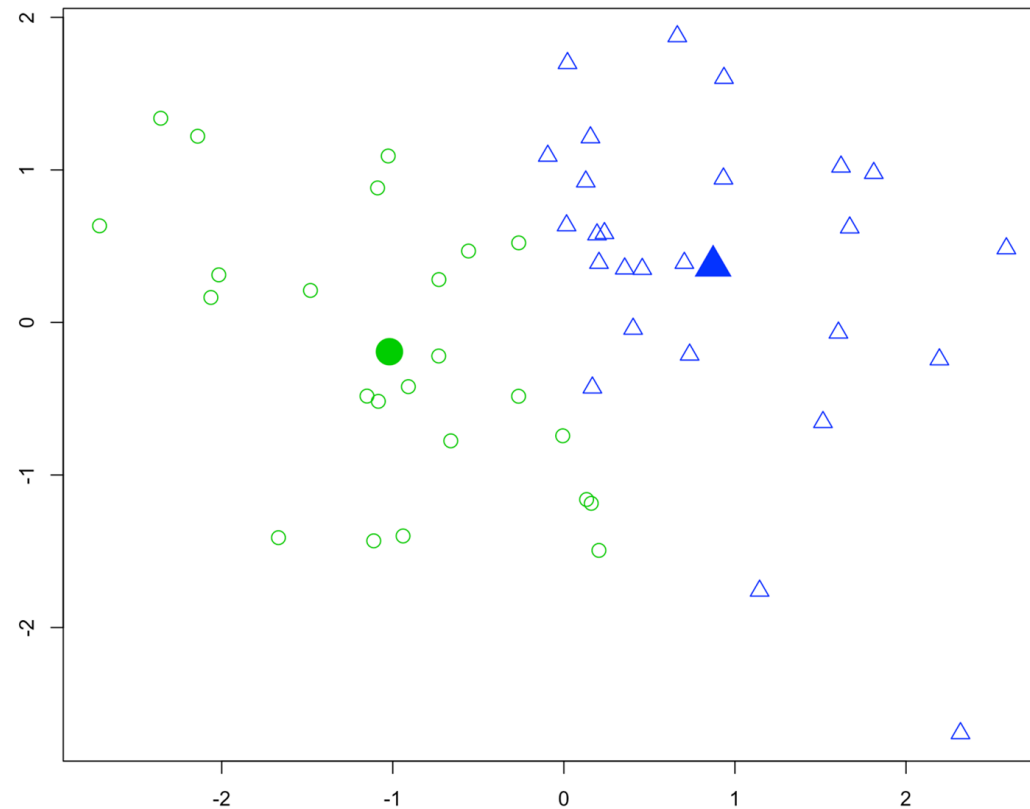
Update centroid and assignment of observations to clusters



► Goal: Partition data in k disjoint subsets

1. Randomly assign k centroids
2. Assign each observation to the cluster whose centroid is closest
3. Compute the new cluster centroids
4. Repeat step 2 and 3 until the result no longer changes
 - As no observations were reassigned to a new cluster during this phase, the k-means algorithm stops.
 - This means the cluster assignments are final when the result no longer changes and thus the clustering has reached a local optimum.

Example: k-Means clustering



► Goal: Partition data in k disjoint subsets

1. Randomly assign k centroids
2. Assign each observation to the cluster whose centroid is closest
3. Compute the new cluster centroids
4. Repeat step 2 and 3 until the result no longer changes

► The centroids do not change anymore

→ The algorithm has converged!

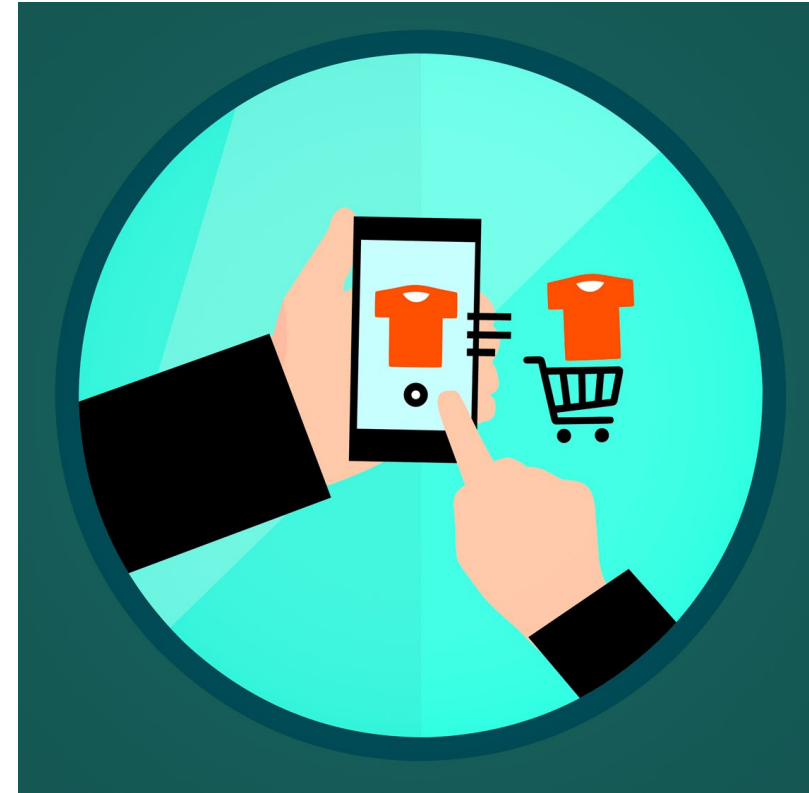
Choosing the appropriate number of clusters for k-Means

- ▶ Choosing the number of clusters requires a delicate balance.
 - ▼ Setting the k to be very large will improve the homogeneity of the clusters.
 - ▼ At the same time, it risks overfitting the data.

- ▶ Guessing the number of clusters
 - ▼ Ideally, you will have some a priori knowledge (that is, a prior belief) about the true groupings, and you can begin applying k-Means using this information.
 - ▼ We will deal with this issue in a later chapter (Finding the right number of clusters)

Running example

- ▶ As a data scientist, you have been hired by a digital commerce company. The company launched a mobile app, and your job is now to gather insights from their customers' data.
- ▶ The company provides you with a data set with the following data:
 - ▼ The time spend on the mobile app
 - ▼ The revenue each customer generated
- ▶ The data is already normalized.

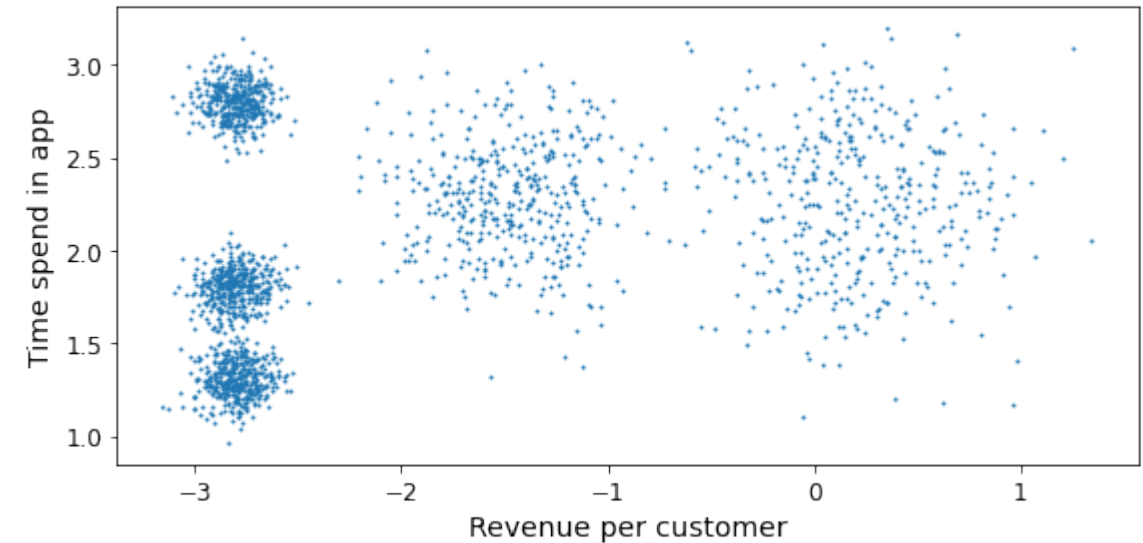


- ▶ Let's first visualize the data.
- ▶ The data visualization shows that there are five cluster types.
 - ▼ It is not always that easy to identify clusters
 - ▼ Most customer data has more than two dimensions, which makes it nearly impossible to manually identify clusters



```
plt.figure(figsize=(8, 4))  
plt.scatter(x=customer["Revenue"], y=customer["Time spend in app"], s=1)  
plt.xlabel("Revenue per customer")  
plt.ylabel("Time spend in app")  
plt.show()
```

- ▶ Let's train a k-Means on this dataset.
 - ▼ It will try to find the center of each cluster and assign each instance to the closest cluster:
- ▶ Note that you have to specify the number of clusters k that the algorithm must find. In this example, it is obvious from looking at the data that k should be set to 5, but in general, it is not that easy.



```
from sklearn.cluster import Kmeans
k=5
kmeans = Kmeans(n_clusters=k, random_state=42) # set random state for reproducibility
y_pred = kmeans.fit_predict(customer)
```

- ▶ We can access the label of each customer by printing out the cluster labels
- ▶ The k-Means instance preserves the labels of the instances it was trained on, available via the `labels_` instance variable
- ▶ We can also access the 5 cluster centers that were estimated:

```
y_pred
```

```
array([4, 0, 1, ..., 2, 1, 0])
```

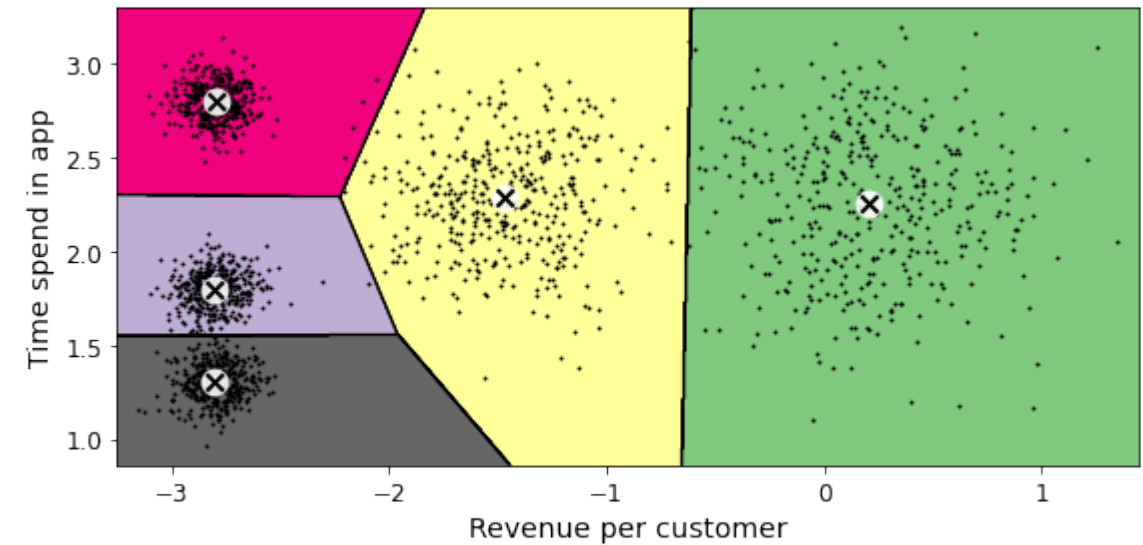
```
y_pred is kmeans.labels_
```

```
True
```

```
kmeans.cluster_centers_
```

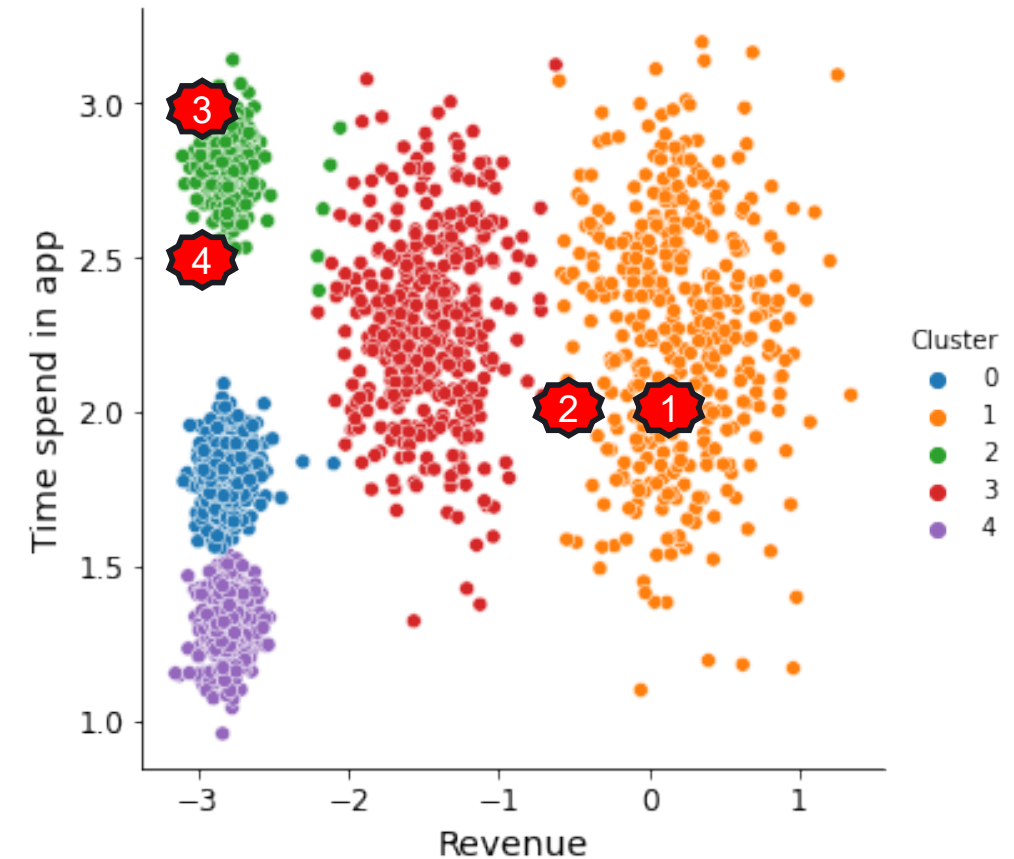
```
array([[ -2.80389616,  1.80117999],  
       [  0.20876306,  2.25513336],  
       [ -2.79290307,  2.79641063],  
       [ -1.46679593,  2.28585348],  
       [ -2.80037642,  1.30082566]])
```


- ▶ We can plot the cluster's decision boundaries.
- ▶ We get a Voronoi tessellation, where each centroid is represented with an (x).
- ▶ We see that the instances near the edges were probably assigned to the wrong cluster, but overall it looks pretty good.



k-Means in Python

- ▶ With the k-Means model, we can also assign new instances of customers to the cluster whose centroid is closest.
- ▶ For instance, the following customers are added as we want to know which cluster they belong to:
 - ▼ Customer 1: Revenue = 0; Time = 2
 - ▼ Customer 2: Revenue = -0.5 ; Time = 2
 - ▼ Customer 3: Revenue = -3; Time = 3
 - ▼ Customer 4: Revenue = -3; Time = 2.5



```
# Create an Array with the new customers
Customer_new = np.array([[0, 2], [-0.5, 2], [-3, 3], [-3, 2.5]])
```

k-Means in Python

► Hard clustering:

- ▼ assign new instances of customers to the cluster whose centroid is closest.

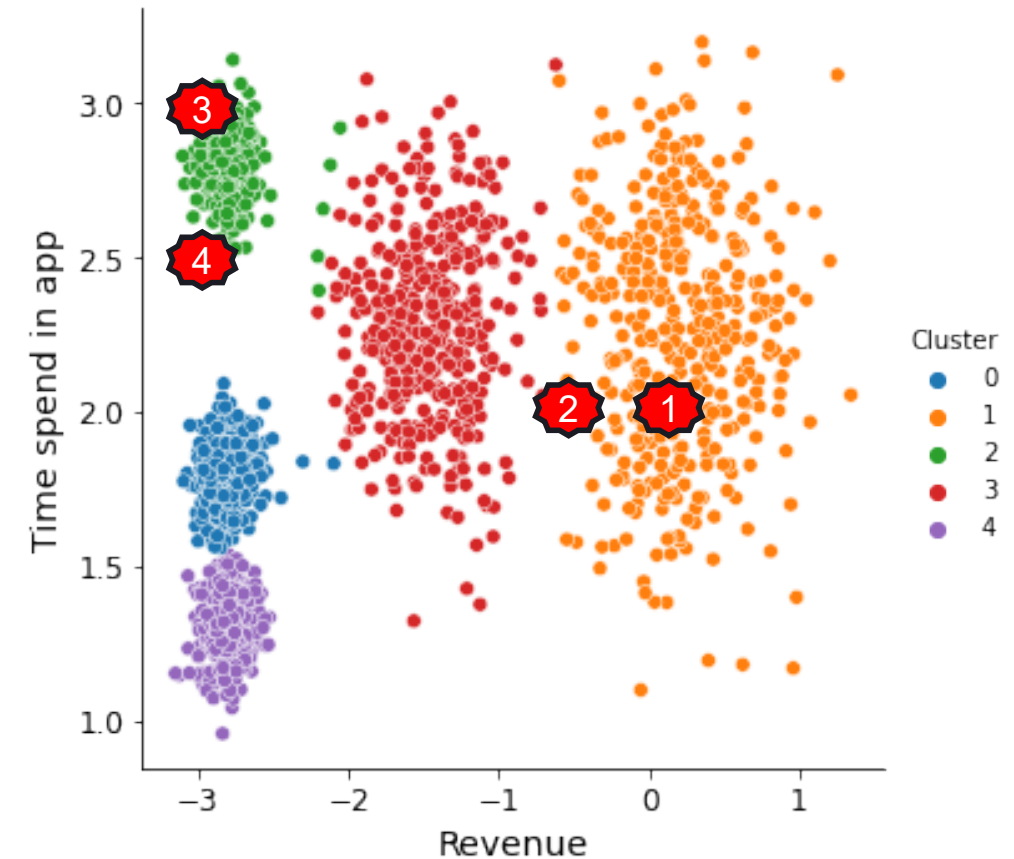
```
kmeans.predict(Customer_new)
```

```
array([1, 1, 2, 2])
```

► Interpretation:

- ▼ Customer 1 belongs to the “orange” cluster (Cluster 1)
- ▼ Customer 2 belongs to the “orange” cluster (Cluster 1)
- ▼ Customer 3 belongs to the “green” cluster (Cluster 2)
- ▼ Customer 4 belongs to the “green” cluster (Cluster 2)

► Note: The cluster start with Cluster 0



k-Means in Python

► Soft clustering:

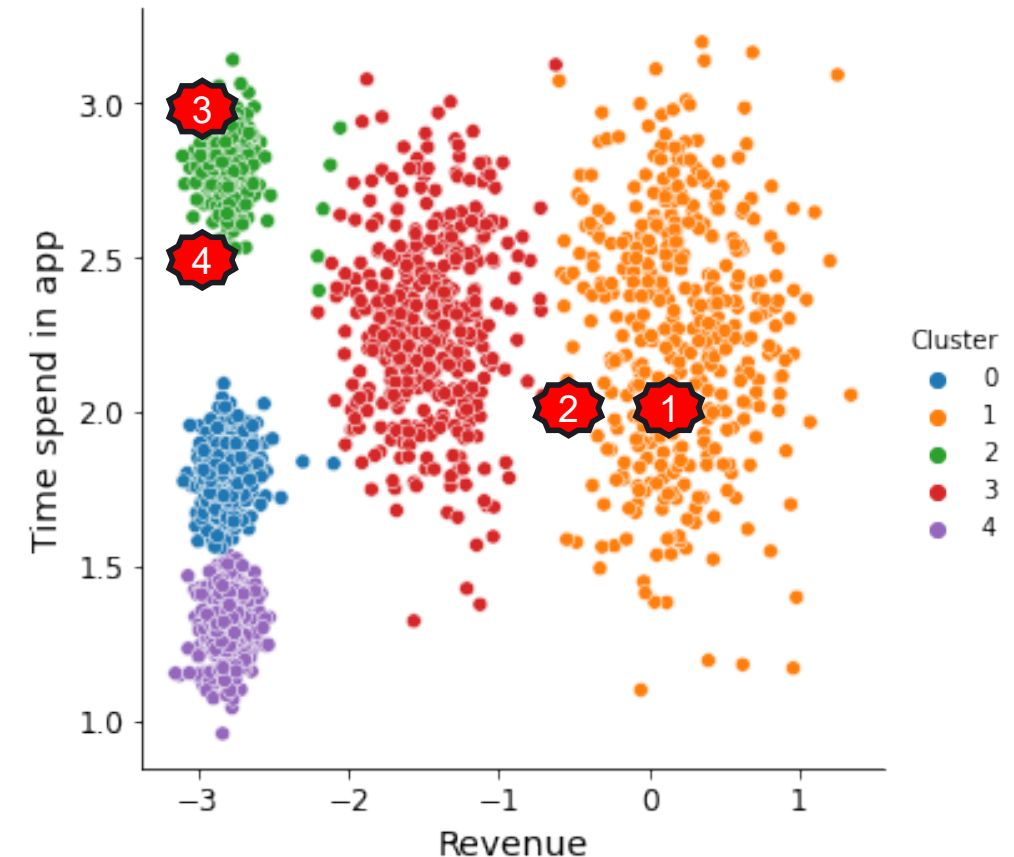
- ▼ it can be useful to give each instance a score per cluster, which is called soft clustering. The score is the distance between the instance and the centroid; conversely, it can be a similarity score.

```
kmeans.transform(Customer_new)
```

```
array([[2.81093633, 0.32995317, 2.9042344 , 1.49439034, 2.88633901],  
       [2.31245906, 0.75341366, 2.42727715, 1.00816991, 2.40428294],  
       [1.21475352, 3.29399768, 0.29040966, 1.69136631, 1.71086031],  
       [0.72581411, 3.21806371, 0.36159148, 1.54808703, 1.21567622]])
```

► Interpretation

- ▼ Customer 1 is located at a distance of 2.810 from the first centroid (“blue cluster”), 0.329 from the second centroid (“orange”), 2.904 from the third centroid (“green”), 1.494 from the fourth centroid (“red”), and 2.886 from the fifth centroid (“purple”).



- Note: The distance of new observations to a cluster can also be used to detect if an observation depicts an outlier. Observations that have a high distance from the existing clusters might be an outlier or an anomaly. This method is often used for “anomaly” / “outlier” / “novelty” detection.

k-Means: advantages and disadvantages

Advantages

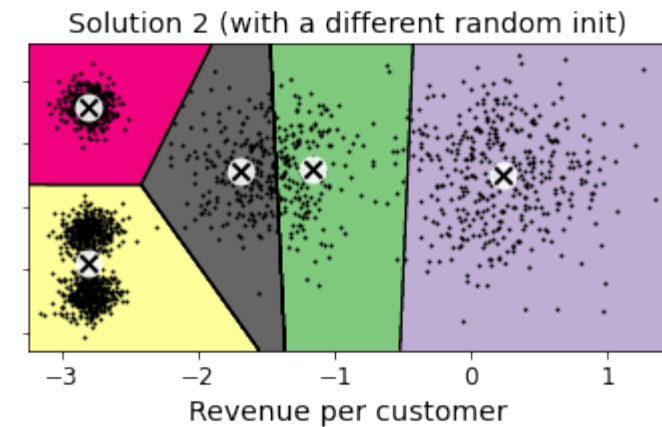
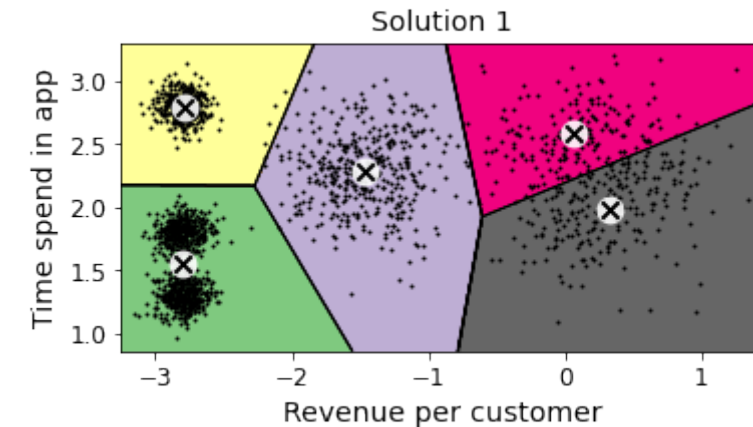
- ▼ Fast computations
- ▼ Uses simple principles for identifying clusters that can be explained in non-statistical terms
- ▼ It is highly flexible and can be adapted to address nearly all of its shortcomings with simple adjustments
- ▼ It is fairly efficient and performs well at dividing the data into useful clusters

Disadvantages

- ▼ The number of clusters must be determined in advance or must be determined experimentally
- ▼ Requires a reasonable guess of how many clusters naturally exist in the data
- ▼ It is less sophisticated than more recent clustering algorithms
- ▼ The algorithm is sensitive to outliers
- ▼ k-Means forms convex (“ball-shaped”) clusters and does not recognize other structures
- ▼ Because it uses an element of random chance (starting centroids), it is not guaranteed to find the optimal set of clusters / the best solution (see next slides)

Disadvantage: k-Means variability

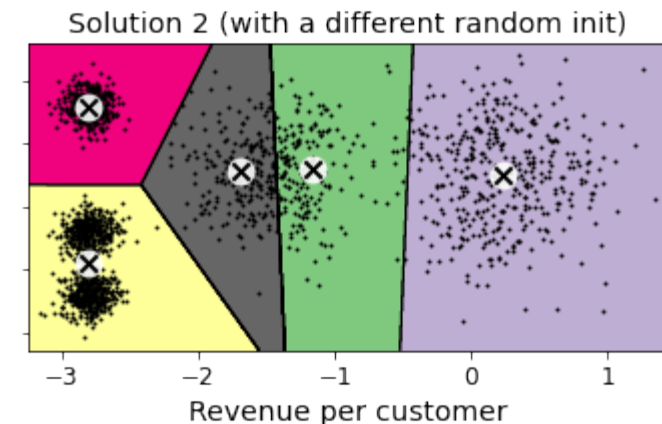
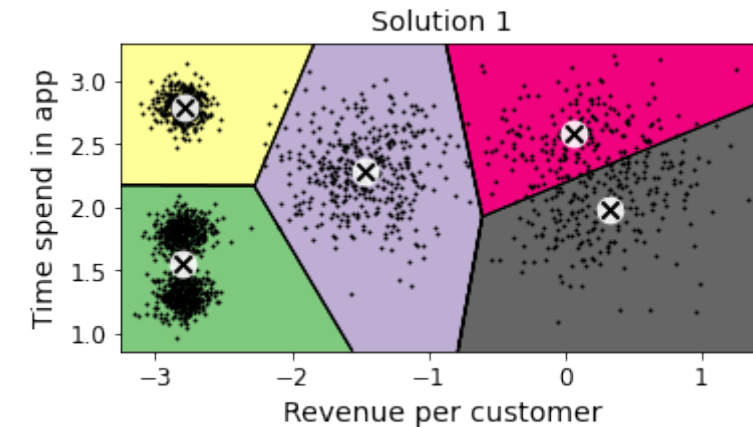
- ▶ k-Means uses an element of random chance (starting centroids), so it is not guaranteed to find the optimal set of clusters / the best solution
- ▶ k-Means can converge to very different solutions, as you can see on the right side
- ▶ Best practice is to run the algorithm several times with different start values and compare the results





Disadvantage: k-Means variability

- ▶ In Python, k-Means runs the algorithm multiple times with different random initializations and keeps the best solution.
- ▶ The number of random initializations is controlled by the `n_init` hyperparameter:
 - ▼ by default, it is equal to 10,
 - ▼ This means that the whole algorithm runs 10 times when you call `fit()`, and Scikit-Learn keeps the best solution.
- ▶ But how exactly does the k-Means algorithm know which solution is the best?
 - ▼ It uses a performance metric
 - ▼ That metric is called the model's inertia
 - ▼ It is the mean squared distance between each instance and its closest centroid ("Within Cluster Sums of Squares")
 - ▼ In the later sections, we discuss in greater detail how to measure the quality of clusters and the role of inertia ([see later slides](#))





Disadvantage: k-Means variability

- We can display the different inertia values with the following commands:

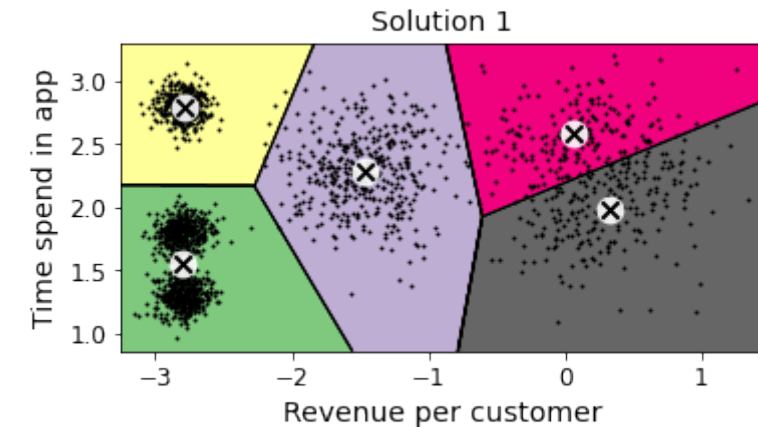
```
kmeans_rnd_init1.inertia_ #for solution 1
```

219.435394427714

```
kmeans_rnd_init2.inertia_ #for solution 2
```

236.82938799181514

- For now, think of inertia as a measure of compactness, the lower the better. More details in the next section inertia ([see later slides](#))
- Interpretation:
 - ▼ Solution 1 on the right side seems to be better than solution 2, because inertia is lower
 - ▼ However, neither of the cluster identifies all clusters perfectly





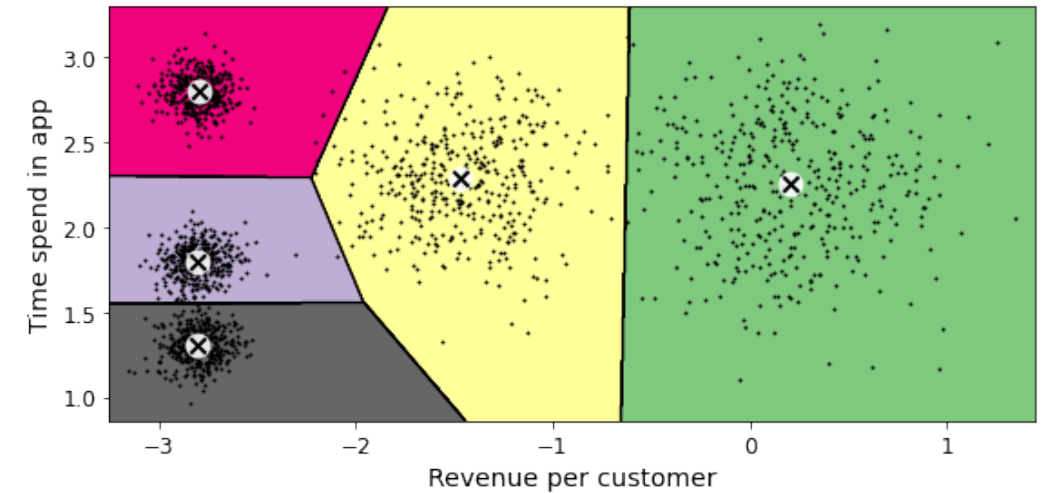
Disadvantage: k-Means variability

- We can display the different inertia values with the following commands:

```
kmeans.inertia_
```

```
211.59853725816828
```

- The first solution on the right side seems to have the best performance because small inertia values are better



Unsupervised Learning: Clustering

- Introduction and overview
- k-Means clustering
- **Finding the right number of clusters**
- Scale issues
- Other cluster algorithms

Evaluation of clustering results

► Meaningfulness and interpretability:

- ▼ Can the cluster be interpreted?
- ▼ Is the result meaningful?
- ▼ Is there a plausible interpretation behind the clusters?

► Internal validation

- ▼ For clustering we do not have labeled data (i.e., which observation belongs to which cluster) to evaluate how “good” our results are
- ▼ The internal validation is therefore based on distance metrics
 - Elbow Plot (to identify the “right” number of clusters)
 - Silhouette’s Index (to identify the “right” number of clusters)
 - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html#sklearn.metrics.silhouette_score
 - Other (not covered in this course)
 - Calinski-Harabasz Index
<https://scikit-learn.org/stable/modules/clustering.html#calinski-harabasz-index>
 - Davis-Bouldin-Index
<https://scikit-learn.org/stable/modules/clustering.html#davies-bouldin-index>

► External validation

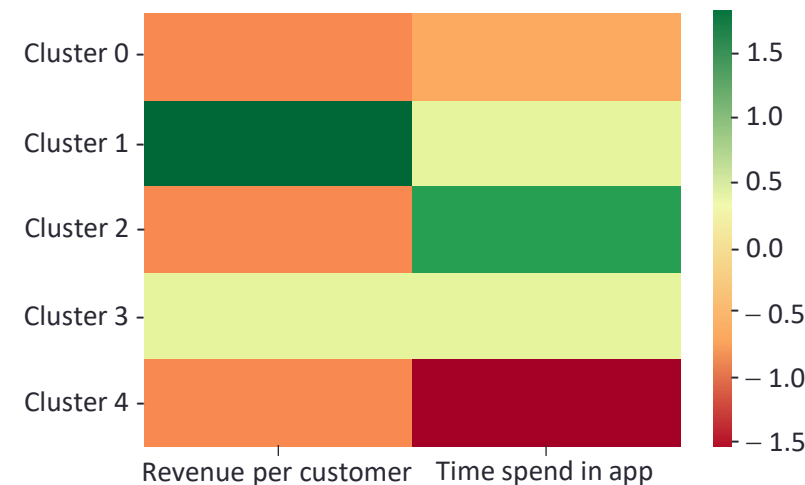
- ▼ External validation, such as done in classification, can be used when we have labeled data (i.e. we know what clusters exist and which observation belongs to which cluster)
- ▼ Measures:
 - Hulbert’s Correlation (not covered in this course)
 - The Jaccard score (not covered in this course)

Meaningfulness and interpretability: heatmap

- ▶ To visualize the results of clustering, heatmaps are often used
 - ▼ Heat maps allow us to simultaneously visualize clusters of samples and features
 - ▼ Make it easier to interpret the results
 - ▼ A heatmap is easier to read if all attributes are measured on the same scale (one can use the sklearn StandardScaler for this)

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(kmeans.cluster_centers_)
scaledClusterCenters = scaler.transform(kmeans.cluster_centers_) # Scale the data

import seaborn as sns
sns.heatmap(scaledClusterCenters, annot=True, fmt='.2f', cmap='RdYlGn',
            xticklabels=x_labels, yticklabels=y_labels)
```



Internal validation: inertia

► Inertia:

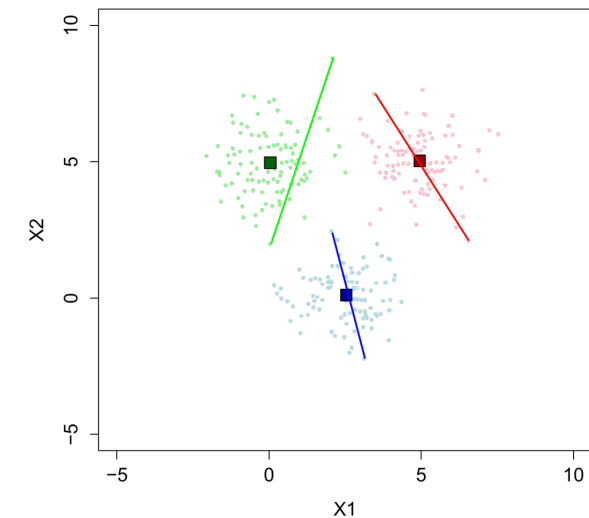
- ▼ It measures the within-cluster sums of squares
- ▼ Reflects a measure of compactness

► Lower values are better, because this implies that points within one cluster are very close to each other

```
kmeans.inertia_
```

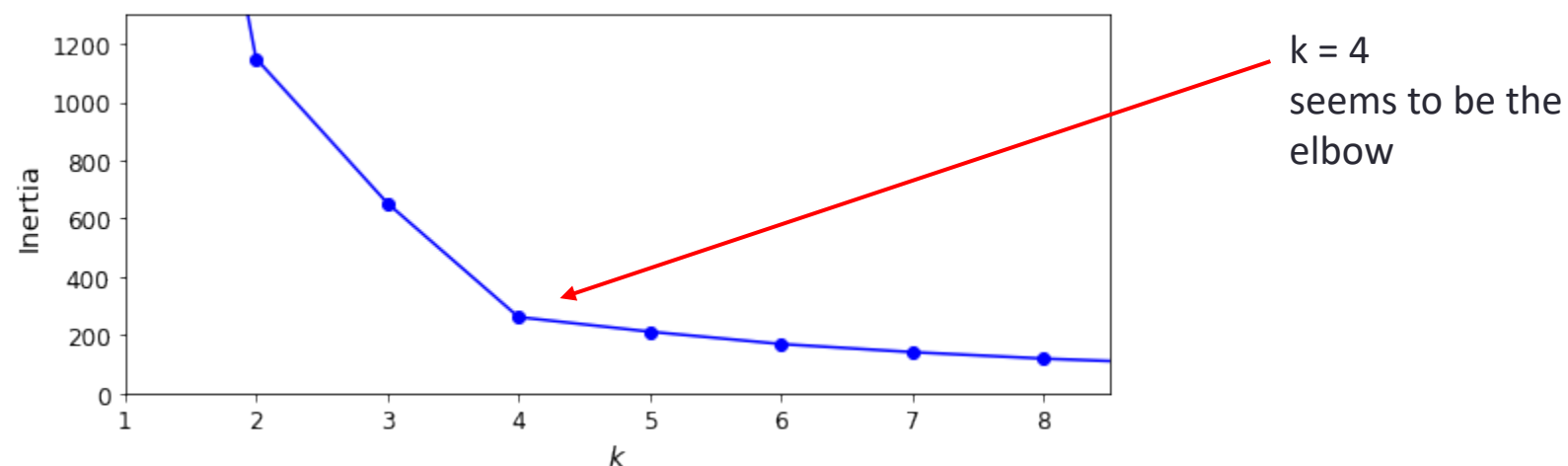
► Example:

- ▼ Visual representation of inertia of three different clusters



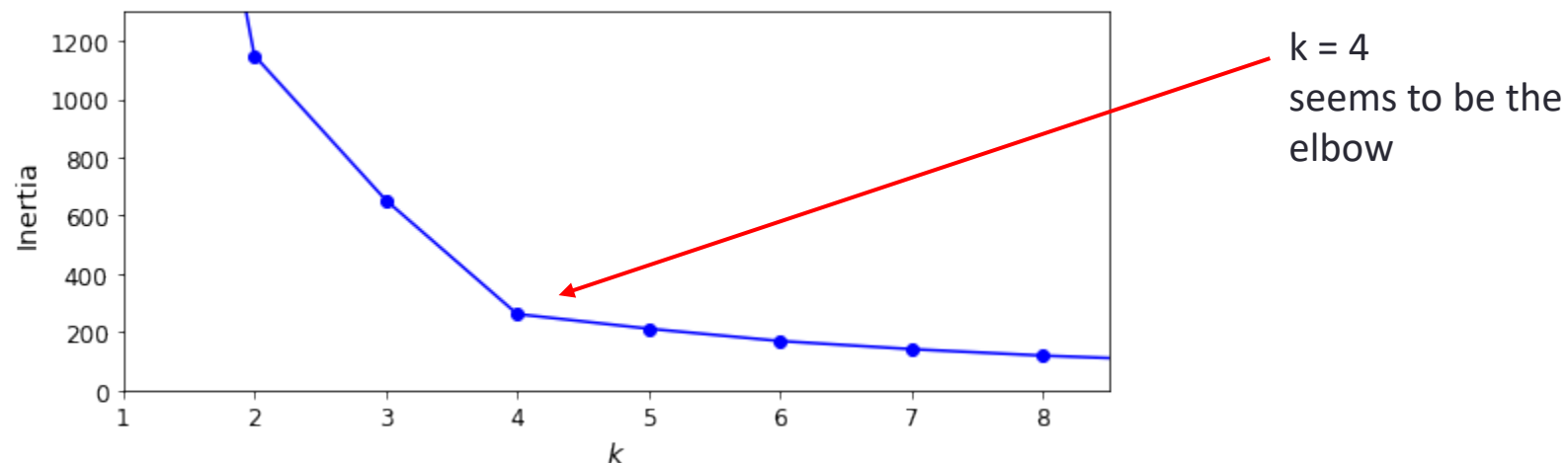
Internal validation: inertia and elbow plot

- ▶ The number (k) of clusters needs to be determined
 - ▼ Goal: Find k that minimizes inertia
 - ▼ Problem: inertia keeps decreasing as k increases!
 - ▼ Solution: find the k value where inertia starts decreasing slowly
- ▶ The inertia may show an inflection point (or “elbow”) at the correct choice of parameter
- ▶ Plotting the validation measure and the number of clusters
 - ▼ The X-axis indicates the number of clusters
 - ▼ The Y-axis illustrates the inertia



Internal validation: inertia and elbow plot (notes)

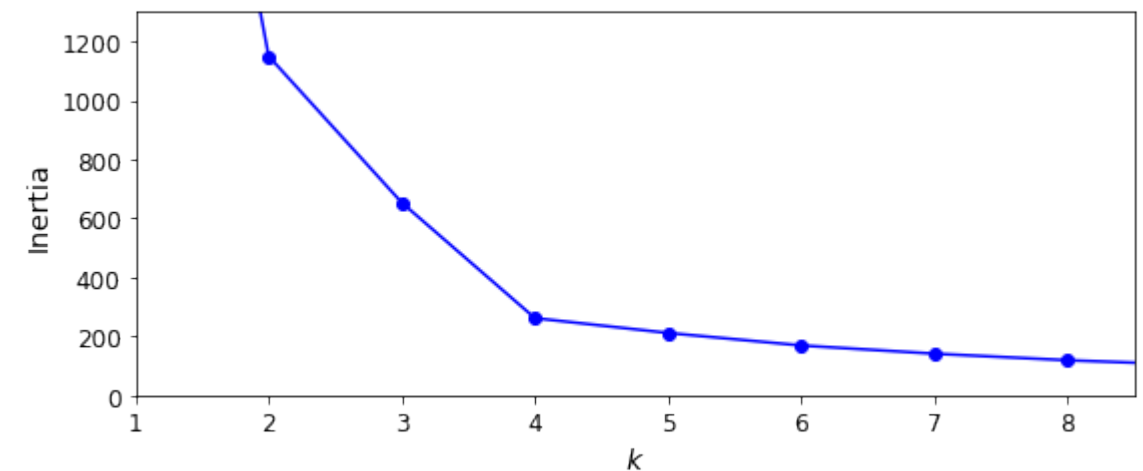
- ▶ Consider the case of k-means clustering where the parameter being tuned is the number of clusters k .
 - ▼ In such a case, inertia measure will always reduce with the number of clusters, though it will reduce at a sharply lower rate after the inflection point.
- ▶ In the visualized example the heterogeneity will continue to decrease with more clusters.
 - ▼ To identify this inflection point (which looks like an elbow), we apply the elbow method. The elbow method gauges how the heterogeneity within the cluster changes for various values of k .
- ▶ To find a “good” number of k , we take a look at the trend of the heterogeneity for a different number of clusters.
 - ▼ The number of clusters where the decrease is no longer decisive can be seen as a good number of k .
- ▶ Additional information:
 - ▼ For a very thorough review of the vast assortment of measures of cluster performance, have a look at Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of intelligent information systems*, 17(2), 107-145.



Internal validation: elbow plot in Python

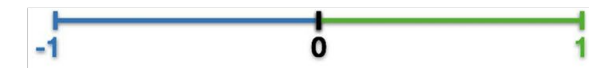
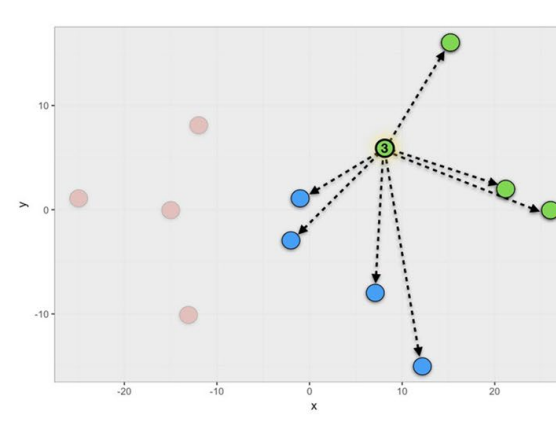
```
kmeans_per_k = [KMeans(n_clusters=k, random_state=42).fit(customer) for k in range(1, 10)]  
inertias = [model.inertia_ for model in kmeans_per_k]  
  
plt.figure(figsize=(8, 3.5))  
plt.plot(range(1, 10), inertias, "bo-")  
plt.axis([1, 8.5, 0, 1300])  
plt.show()
```

- The elbow plot shows that $k = 4$ or $k = 5$ could be a good choice.



Internal validation: silhouette method

- ▶ The Silhouette method will evaluate the quality of clusters by how well each point fits within a cluster, maximizing average "silhouette" width.
- ▶ Silhouette width ranges from -1 to 1 for each observation in your data and can be interpreted as follows:
 - ▼ Values close to 1 suggest that the observation is well matched to the assigned cluster
 - ▼ Values close to 0 suggest that the observation is borderline matched between two clusters
 - ▼ Values close to -1 suggest that the observations may be assigned to the wrong cluster



- ▶ The silhouette coefficient
 - ▼ is the average of each observations silhouette width
 - ▼ Large positive values indicate highly separated clustering,
 - ▼ Negative values are indicative of some level of "mixing" of data points from different clusters.

1: Well matched to cluster
0: On border between two clusters
-1: Better fit in neighboring cluster

Internal validation: silhouette method

- ▶ The Silhouette coefficient will evaluate the quality of clusters by how well each point fits within a cluster, maximizing the average "silhouette" width.
 - ▼ It finds how well each data point is classified.
 - ▼ The plot of the silhouette score helps us to visualize and interpret how well data points are tightly grouped within their clusters and separated from others.
 - ▼ It helps us to evaluate the number of clusters. Its score ranges from -1 to +1.
 - ▼ A positive value indicates a well-separated cluster and a negative value indicates incorrectly assigned data points.
 - ▼ The more positive the value, the further data points are from the nearest clusters; a value of zero indicates data points that are at the separation line between two clusters.

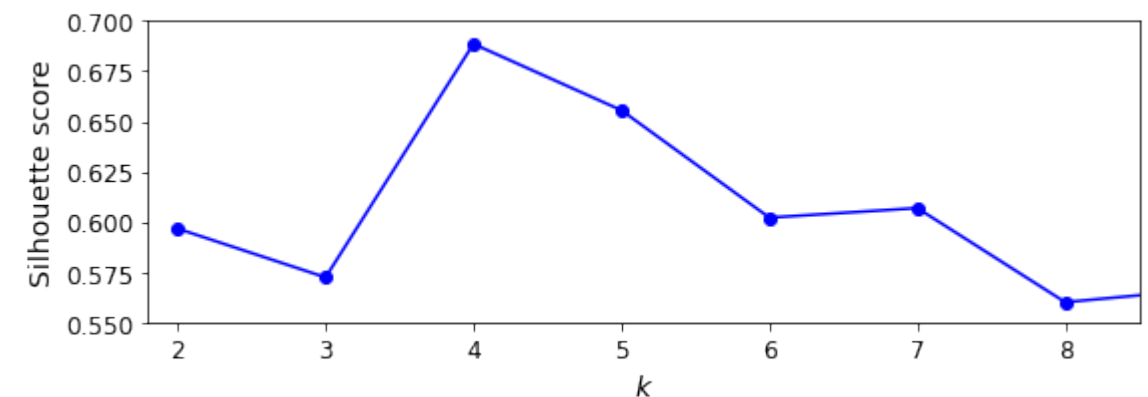
Internal validation: elbow plot in Python

```
from sklearn.metrics import silhouette_score
kmeans_per_k = [Kmeans(n_clusters=k, random_state=42).fit(customer) for k in range(1, 10)]

silhouette_scores = [silhouette_score(customer, model.labels_) for model in kmeans_per_k [1:]]

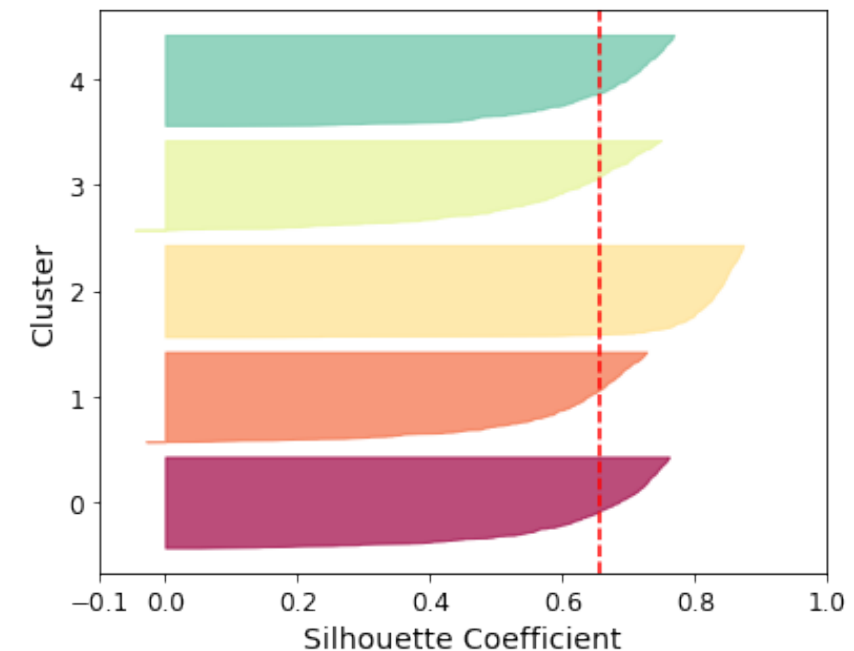
plt.figure(figsize=(8, 3))
plt.plot(range(2, 10), silhouette_scores, "bo-")
plt.show()
```

- ▶ As we can see, this visualization gives a detailed picture
- ▶ It confirms that $k=4$ is a very good choice, but it also underlines the fact that $k=5$ is quite good as well.



Internal validation: silhouette diagram

- ▶ An even more informative visualization is the silhouette diagram
- ▶ For this, we plot every instance's silhouette coefficient, sorted by the cluster they are assigned to and by the value of the coefficient.
- ▶ Each diagram contains one knife shape per cluster.
- ▶ The shape's height
 - ▼ Indicates the number of instances the cluster contains,
 - ▼ Represents the sorted silhouette coefficients of the instances in the cluster (wider is better).
- ▶ The dashed line
 - ▼ Indicates the mean silhouette coefficient.
 - ▼ When most of the instances in a cluster have a lower coefficient than this score (i.e., if any of the instances stop short of the dashed line, ending to the left of it), then the cluster is rather bad since this means its instances are much too close to other clusters.



Internal validation: silhouette diagram in Python

```

from sklearn.metrics import silhouette_samples
from matplotlib.ticker import FixedLocator, FixedFormatter

plt.figure(figsize=(11, 9))
kmeans_per_k = [KMeans(n_clusters=k, random_state=42).fit(customer) for k in range(1, 10)]

for k in (3, 4, 5, 6):
    plt.subplot(2, 2, k - 2)

    y_pred = kmeans_per_k[k - 1].labels_
    silhouette_coefficients = silhouette_samples(customer, y_pred)

    padding = len(customer) // 30
    pos = padding
    ticks = []
    for i in range(k):
        coeffs = silhouette_coefficients[y_pred == i]
        coeffs.sort()

        color = mpl.cm.Spectral(i / k)
        plt.fill_betweenx(np.arange(pos, pos + len(coeffs)), 0, coeffs,
                          facecolor=color, edgecolor=color, alpha=0.7)

        ticks.append(pos + len(coeffs) // 2)
        pos += len(coeffs) + padding

    plt.gca().yaxis.set_major_locator(FixedLocator(ticks))
    plt.gca().yaxis.set_major_formatter(FixedFormatter(range(k)))
    if k in (3, 5):
        plt.ylabel("Cluster")

    if k in (5, 6):
        plt.gca().set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])
        plt.xlabel("Silhouette Coefficient")
    else:
        plt.tick_params(labelbottom=False)

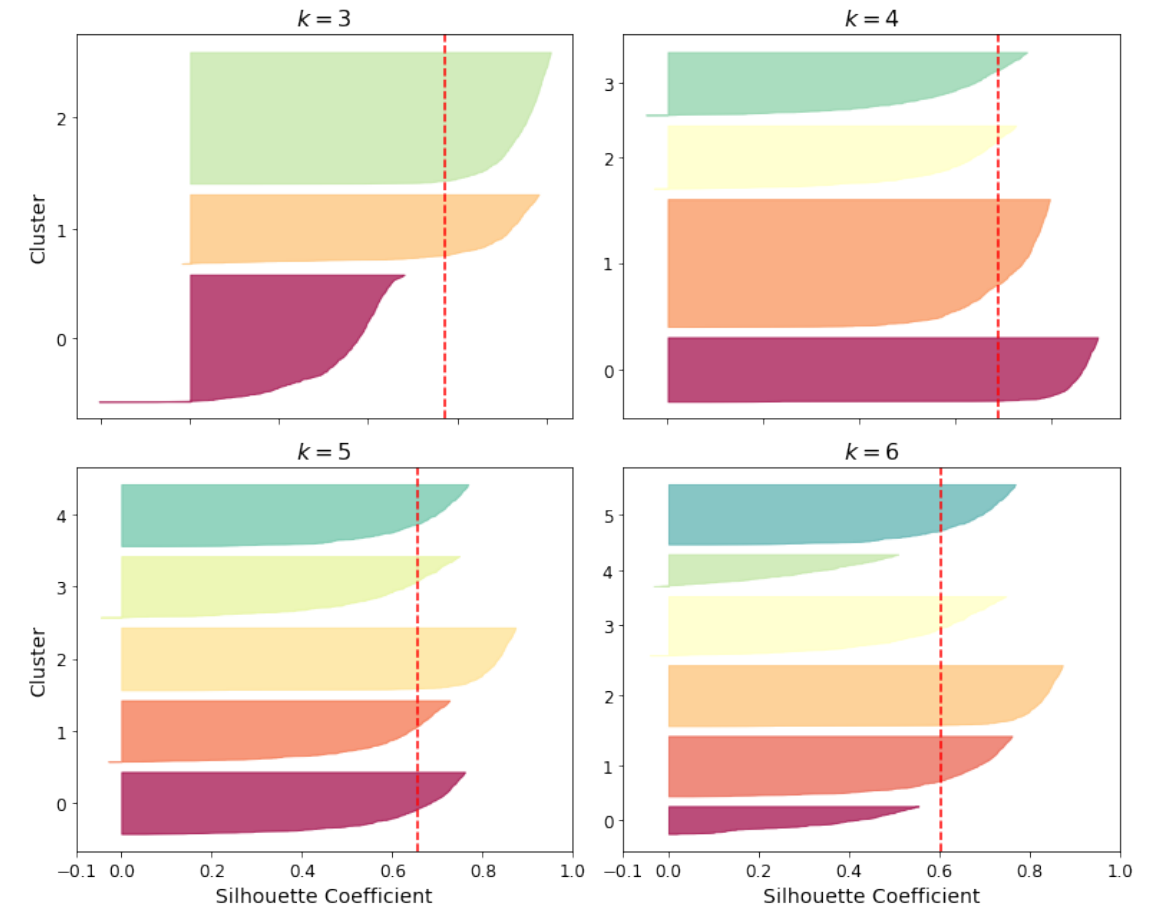
    plt.axvline(x=silhouette_scores[k - 2], color="red", linestyle="--")
    plt.title("$k={}$".format(k), fontsize=16)

plt.show()

```

Internal validation: silhouette diagram

- ▶ We can see that when $k = 3$ and when $k = 6$, we get bad clusters.
 - ▼ Some of the clusters are below the mean silhouette coefficient
 - ▼ Some of the clusters are very small
- ▶ But when $k = 4$ or $k = 5$, the clusters look pretty good: most instances extend beyond the dashed line, to the right, and closer to 1.0.
 - ▼ When $k = 4$, the cluster at index 1 (the third from the top) is rather big.
 - ▼ When $k = 5$, all clusters have similar sizes.
- ▶ Even though the overall silhouette score from $k = 4$ is slightly greater than for $k = 5$, it seems like a good idea to use $k = 5$ to get clusters of similar sizes.



Unsupervised Learning: Clustering

- Introduction and overview
- k-Means clustering
- Finding the right number of clusters
- **Scale issues**
- Other cluster algorithms

- ▶ Problem: Data on different scales can cause undesirable results in clustering methods
- ▶ Solution: Scale data so that features have the same mean and standard deviation (standardization of data)
 - ▼ Subtract the mean of a feature from all observations
 - ▼ Divide each feature by the standard deviation of the feature
 - Normalized features have a mean of zero and a standard deviation of one

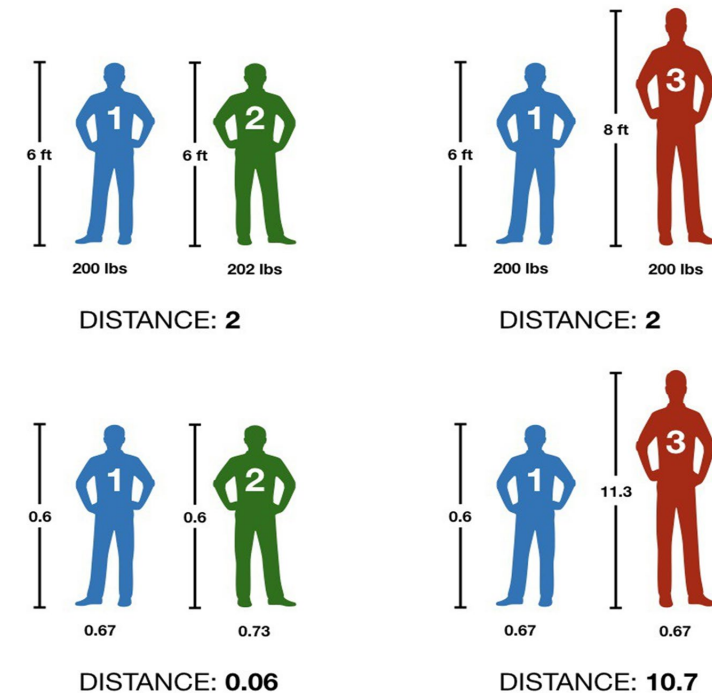
- ▶ Example

- ▼ Distance of two persons:

- Person 1 (200 lbs) vs. person 2 (202 lbs) → distance 2
 - Person 1 (6 ft) vs. person 3 (8ft) → distance 2
 - Same distance but not comparable because of different scales

- ▼ Solution: Scale data by dividing by the standard deviation

- Person 1 (0.57) vs. person 2 (0.73) → distance 0.06
 - Person 1 (0.67) vs. person (10.7) → distance 10.7



- ▶ In Python:

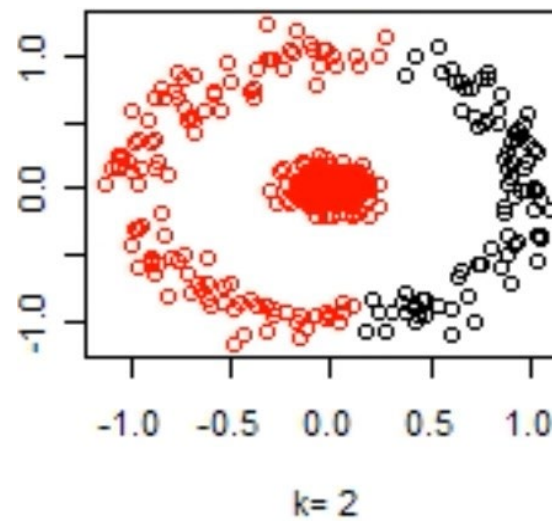
<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

Unsupervised Learning: Clustering

- Introduction and overview
- k-Means clustering
- Finding the right number of clusters
- Scale issues
- **Other cluster algorithms**

Disadvantages of k-Means clustering

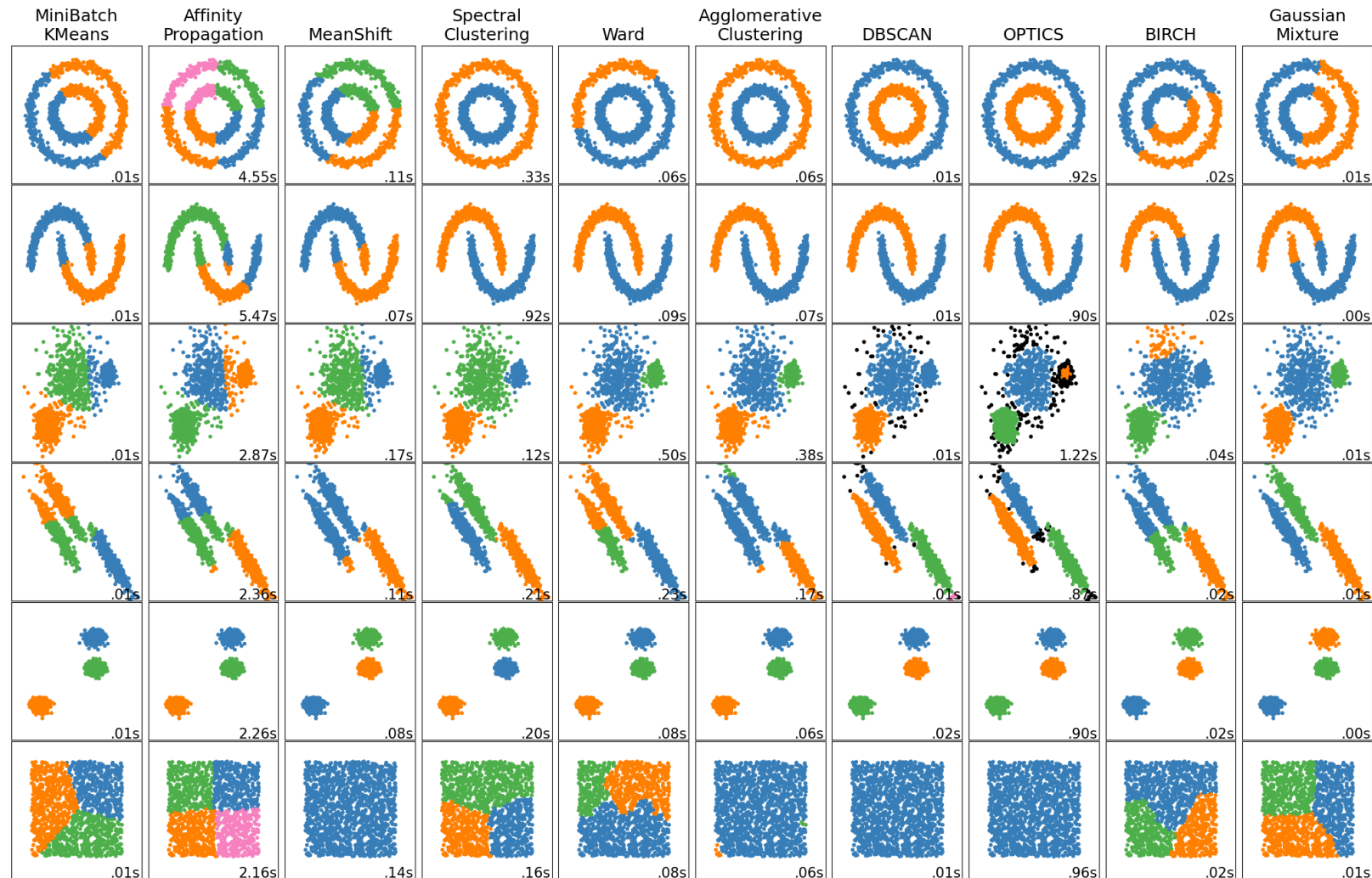
- ▶ Clusters should be similar in size since the algorithm partitions the data in the “middle” between two cluster centers
- ▶ K-means forms convex clusters and does not recognize other structures



- ▶ There exist various other cluster algorithms that can be used

A comparison of the clustering algorithms in scikit-learn

► URL: <https://scikit-learn.org/stable/modules/clustering.html#affinity-propagation>



A comparison of the clustering algorithms in scikit-learn

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large n_samples, medium n_clusters with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters, inductive	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry, inductive	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry, inductive	Distances between points
Spectral clustering	number of clusters	Medium n_samples, small n_clusters	Few clusters, even cluster size, non-flat geometry, transductive	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, transductive	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, non Euclidean distances, transductive	Any pairwise distance

A comparison of the clustering algorithms in scikit-learn

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
DBSCAN	neighborhood size	Very large n_samples, medium n_clusters	Non-flat geometry, uneven cluster sizes, outlier removal, transductive	Distances between nearest points
OPTICS	minimum cluster membership	Very large n_samples, large n_clusters	Non-flat geometry, uneven cluster sizes, variable cluster density, outlier removal, transductive	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation, inductive	Mahalanobis distances to centers
BIRCH	branching factor, threshold, optional global clusterer.	Large n_clusters and n_samples	Large dataset, outlier removal, data reduction, inductive	Euclidean distance between points

A comparison of the clustering algorithms in scikit-learn

► **Note:**

- Non-flat geometry clustering is useful when the clusters have a specific shape, i.e. a non-flat manifold, and the standard Euclidean distance is not the right metric. This case arises in the two top rows of the figure above.
- Gaussian mixture models, useful for clustering, are described in another chapter of the documentation dedicated to mixture models. KMeans can be seen as a special case of the Gaussian mixture model with equal covariance per component.
- Transductive clustering methods (in contrast to inductive clustering methods) are not designed to be applied to new, unseen data.

You can learn all this in an interactive online course!

<https://app.datacamp.com/learn/courses/cluster-analysis-in-python>



Unsupervised Learning: Learning objectives

After this lecture, students shall be able to...

- name problem classes that can be solved using cluster analysis,
- explain the functional principle of K-Means clustering
- explain how a good number of clusters can be determined