

# Modeling

---

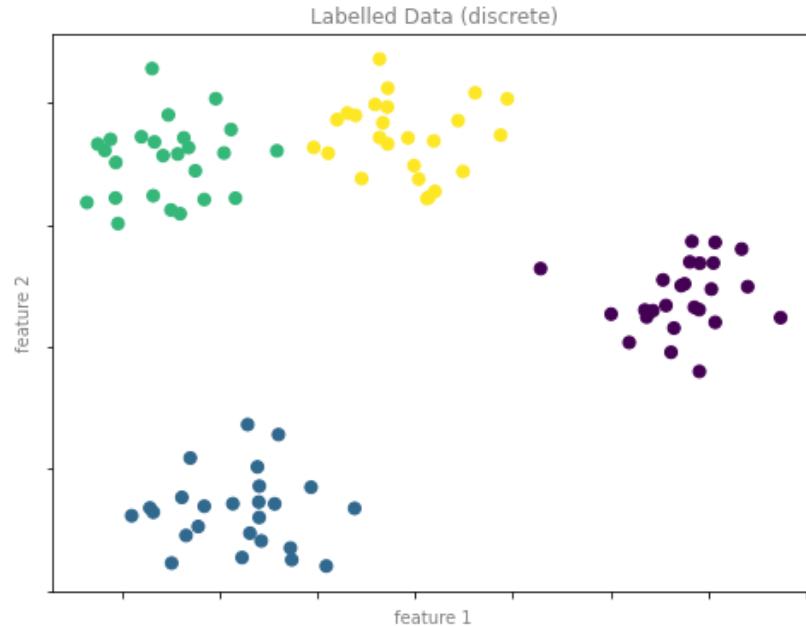
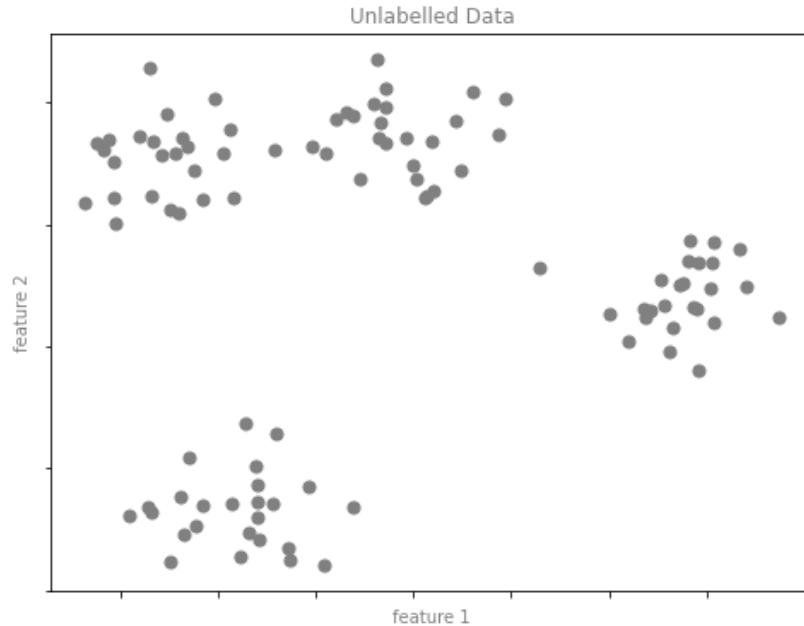
Unsupervised Learning: Clustering

# Agenda

---

- Clustering im Allgemeinen
- k-means Clustering: generell und in `sklearn`
- Expectation-Maximization Algorithmus
- k-means im Detail
- Nachteile des EM-Algorithmus
- Validierung von Clustering
- Variationen k-means
- Dichtebasiertes Clustering: DBSCAN

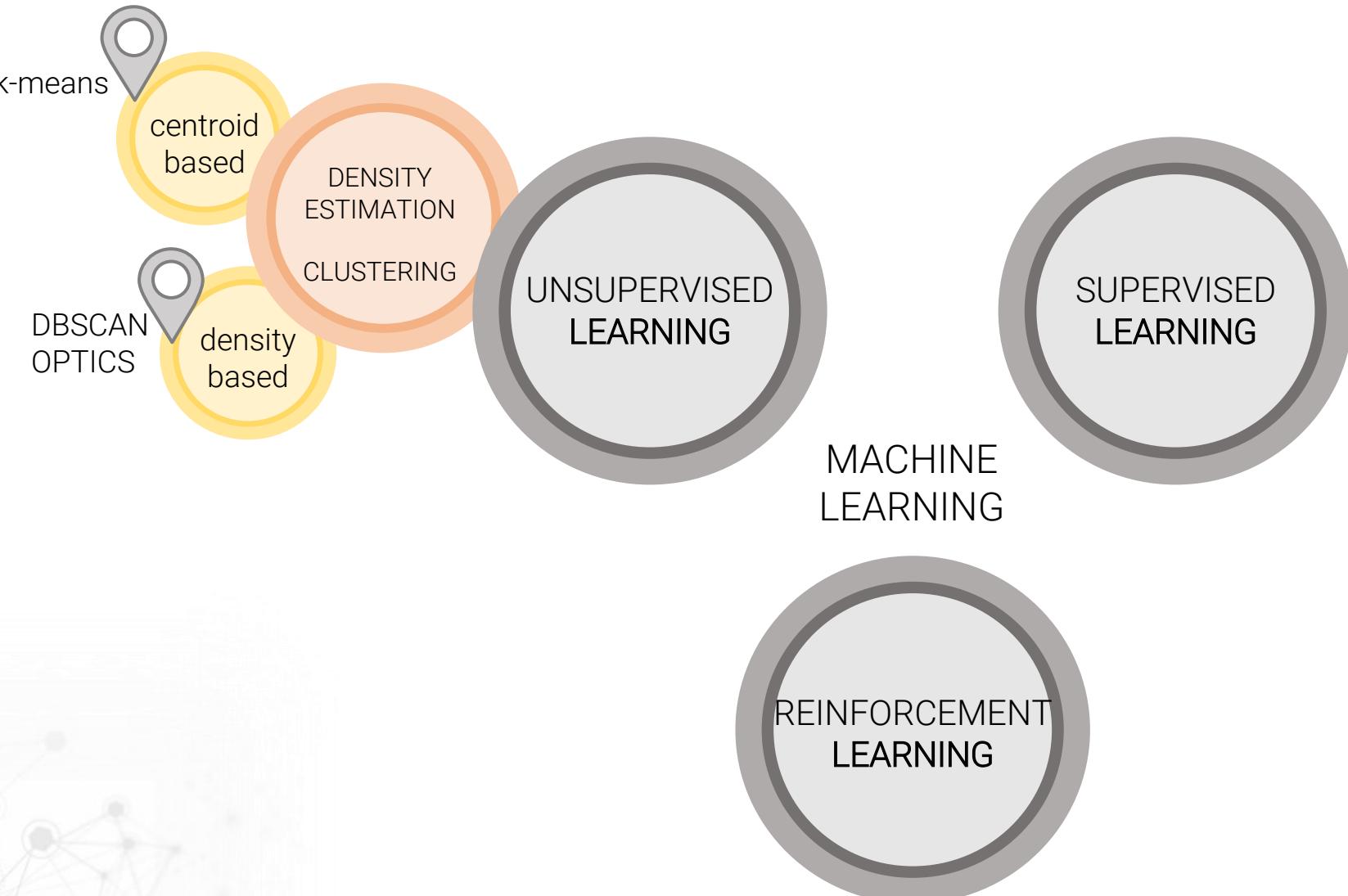
# Was sehen Sie?



*So what?*

- Ihr visueller Apparat entdeckt automatisch und ohne Mühe Gruppierungen bzw. Struktur in Daten
- Wie bringen wir das Algorithmen bei?<sup>1</sup>

# SVM: wo befinden wir uns?

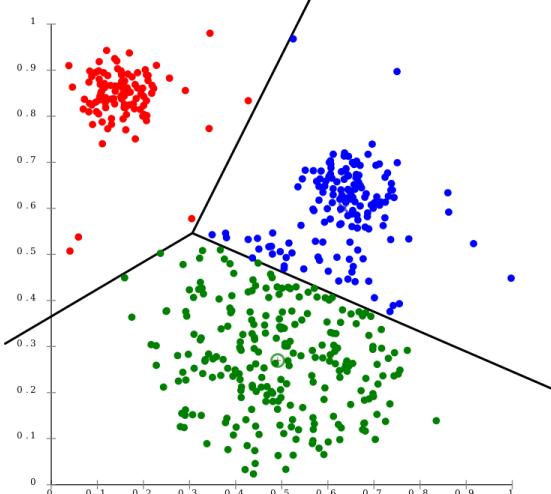


# Clustering im Allgemeinen

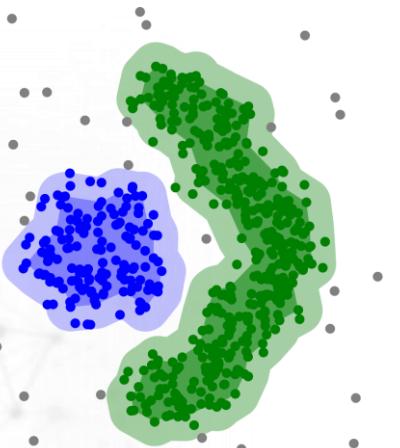


Was denken Sie?

Partitionierend



Dichtebasiert



Wie würden Sie mit eigenen Worten beschreiben, was ein Clustering bewirkt bzw. durchführt?



Was denken Sie?

Was heißt hier optimal?

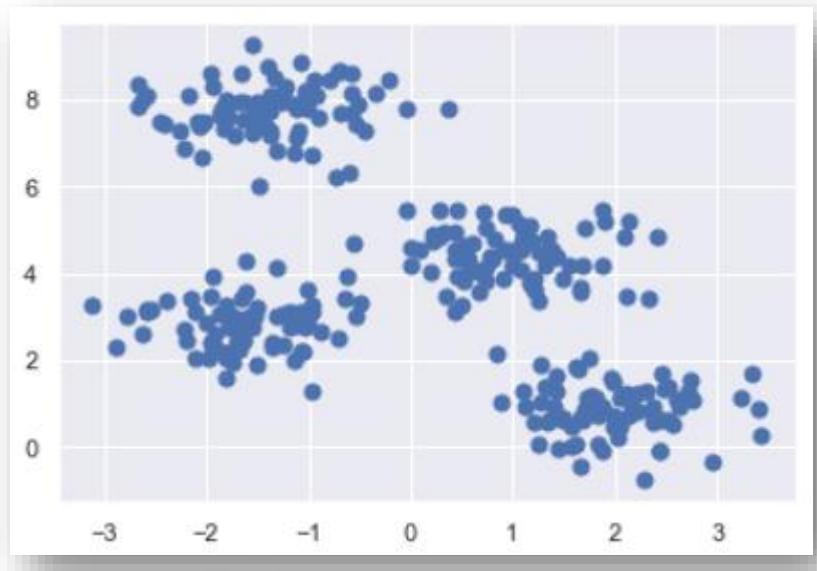
- Clustering Algorithmen versuchen aus den Eigenschaften der Daten eine **optimale Unterteilung in Gruppen** zu erreichen → diskrete Labels
- Es gibt **verschiedene Arten des Clusterings**:
  - Partitionierendes Clustering (z.B. k-means und Varianten)
  - Dichtebasierter Clustering (z.B. DBSCAN, OPTICS)
  - Hierarchisches Clustering
  - Gitterbasierte Verfahren

# k-means: im Allgemeinen



Was denken Sie?

Wie wird ein Cluster definiert?



- Wir beginnen mit dem *k-means* Algorithmus
- Der *k-means* Algorithmus sucht nach einer **vorabbestimmten Anzahl** an Clustern in einem **nicht gelabelten** Datensatz
- **Heuristik** des *k-means* Algorithmus:
  - Der Cluster-Mittelpunkt (== **Centroid**) ist das arithmetische Mittel aller Punkte des jeweiligen Clusters
  - Jeder Datenpunkt ist **näher** an seinem eigenen Centroid als an anderen



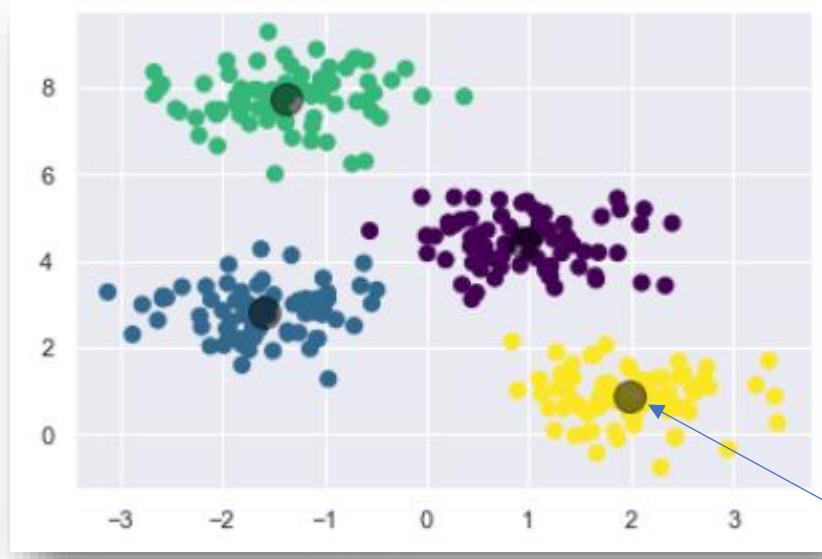
*So what?*

Unsere Leitfrage: Wie macht der *k-means* Algorithmus das, was wir „mit dem Auge“ mühelos können?

```
1 X, y_true = make_blobs(n_samples=300, centers=4,  
2                         cluster_std=0.60, random_state=0)  
3 plt.scatter(X[:, 0], X[:, 1], s=50);
```

# k-means: sklearn

```
1 from sklearn.cluster import KMeans  
2 kmeans = KMeans(n_clusters=4)  
3 kmeans.fit(X)  
4 y_kmeans = kmeans.predict(X)
```



Centroid

```
1 plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')  
2 centers = kmeans.cluster_centers_  
3 plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);
```

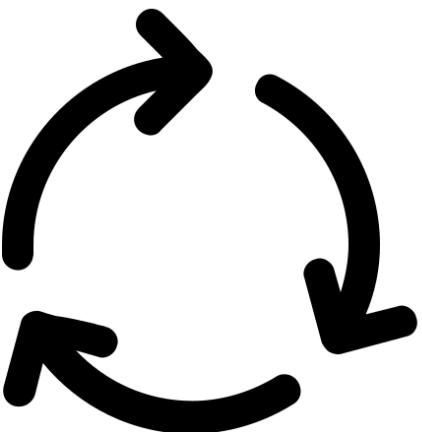
- In sklearn weist der k-means Algorithmus die gleiche Struktur wie andere Modelle auf
- Als wichtiger Zusatz bei der Instanziierung des Modells müssen wir die **Anzahl an Clustern** angeben
- Die **Zugehörigkeit** bzw. **Labels** der Datenpunkte (auch neuer bzw. ungesehener Datenpunkte) zu den jeweiligen Centroids erhalten wir über die `.predict()`-Methode
- Die erlernten Centroids stecken im `.cluster_centers_-Attribut` des Modells



Was denken Sie?

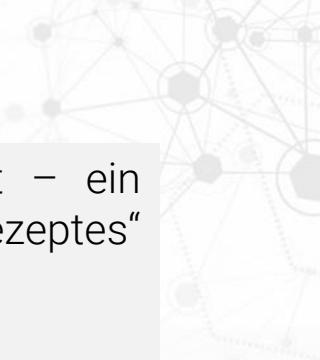
Wie findet der Algorithmus die Centroids ohne alle möglichen Kombinationen explizit zu evaluieren?

# Einschub: Expectation-Maximization



- Die Heuristik der vorletzten Folie wird umgesetzt – ein „Rezept“ entsteht → die allgemeine Form dieses „Rezeptes“ ist der sog. **Expectation-Maximization-Algorithmus**
- Der EM-Algorithmus findet in vielen Machine Learning Modellen Anwendung
- Es handelt sich um einen **iterativen** Algorithmus
- **Generelle Idee:**
  1. Initialisiere eine **zufällige** Konfiguration des Modells
  2. Führe folgende zwei Schritte abwechselnd bis **Konvergenz** aus:
    - i. **E(xpectation)-Schritt:** Zuordnung der Daten zu den einzelnen Teilen des Modells
    - ii. **M(aximization)-Schritt:** Parameter an die neueste Zuordnung anpassen
- Konvergenz: findet keine **wesentliche** Verbesserung mehr statt → Abbruch

# Einschub: Expectation-Maximization



Was denken Sie?

Was sind die analogen  
Schritte beim k-means?

Initialisiere Centroids zufällig

Weise Datenpunkte zum  
nächsten Centroid zu

Berechne die neuen Centroids als  
arithmetisches Mittel der neuen Datenpunkte



So what?

- *Expectation*: bedeutet beim k-means also „unsere Erwartung“ updaten, zu welchem Cluster die Datenpunkte gehören
- *Maximization*: ein Gütemaß wird maximiert (in unserem Fall durch die Berechnung des Mittelwertes)

- Die Heuristik der vorletzten Folie wird umgesetzt – ein „Rezept“ entsteht → die allgemeine Form dieses „Rezeptes“ ist der sog. *Expectation-Maximization-Algorithmus*

- Der EM-Algorithmus findet in vielen Machine Learning Modellen Anwendung

- Es handelt sich um einen **iterativen** Algorithmus

- **Generelle Idee:**

1. Initialisiere eine **zufällige** Konfiguration des Modells
2. Führe folgende zwei Schritte abwechselnd *bis* Konvergenz aus:

- i. **E(xpectation)-Schritt**: Zuordnung der Daten zu den einzelnen Teilen des Modells
- ii. **M(aximization)-Schritt**: Parameter an die neueste Zuordnung anpassen

- Konvergenz: findet keine **wesentliche** Verbesserung mehr statt → Abbruch

# k-means: Input und Output



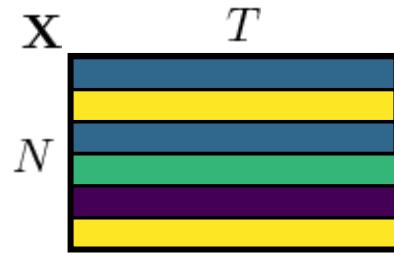
$\mathbf{X}$ : Data matrix with dimensions  $N \times T$

$N$ : number of observations

$T$ : number of features (e.g.  $\bar{x}, \sigma^2, \dots$ )

$k$ : number of clusters to be extracted

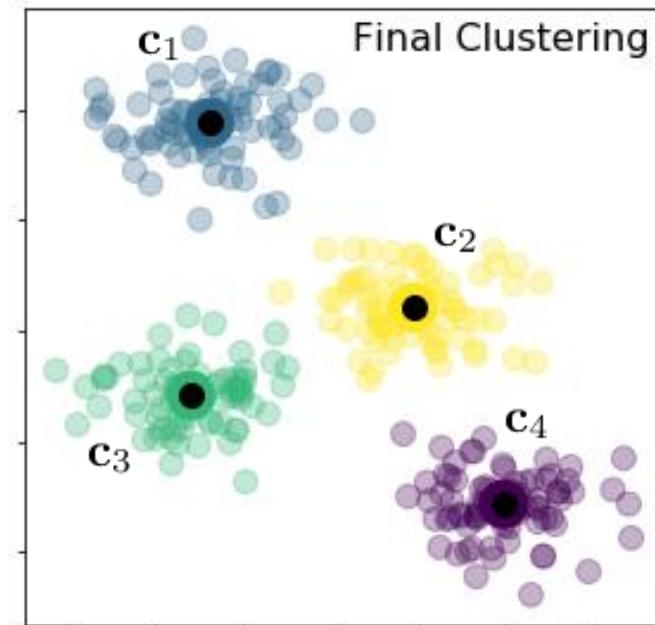
*Input*



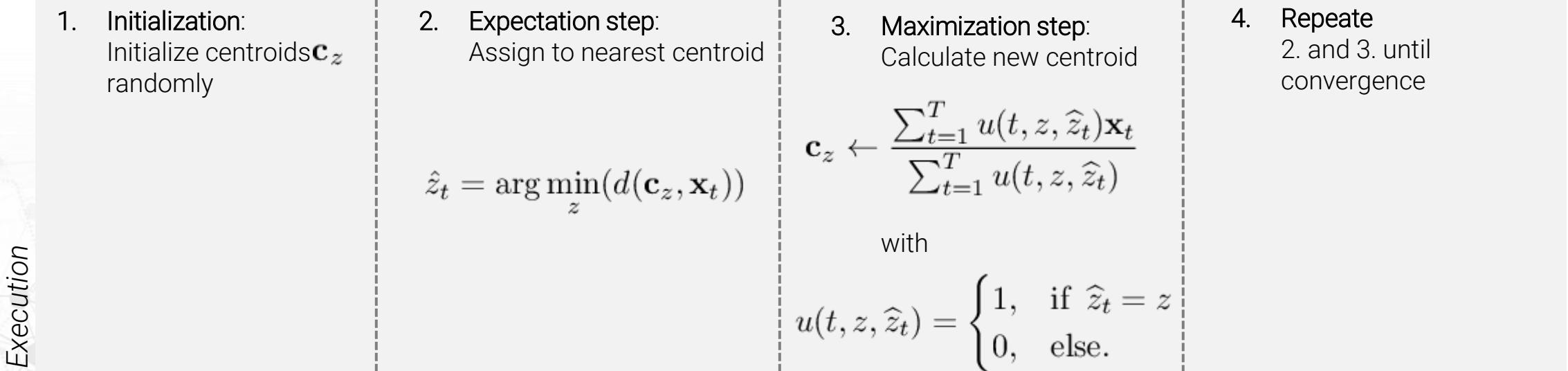
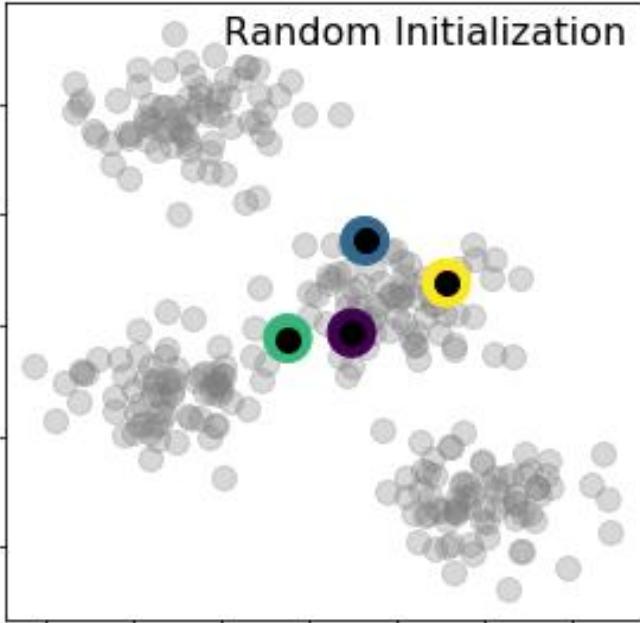
$\mathbf{c}_z$ : cluster centroids with  $z = 1, \dots, k$

$\hat{z}_t$ : assigned cluster index of data point  $\mathbf{x}_t$

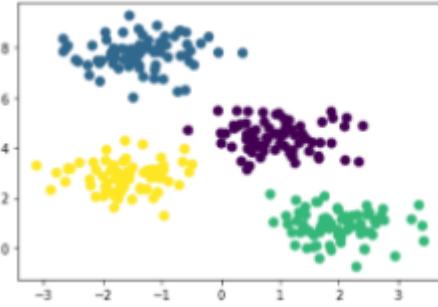
*Output*



# k-means: Ausführung



# Beispiel: k-means from Scratch



Der k-means Algorithmus ist so anschaulich, dass wir diesen Algorithmus "from Scratch" - also im Detail - nachprogrammieren können. Unsere Aufgabe in diesem Beispiel ist es also, den k-means Algorithmus Schritt für Schritt zu implementieren. Schreiben Sie also eine Funktion, die als Eingabeargumente die Daten und die Anzahl zu extrahierender Cluster aufnimmt - und uns die Centroids und Labels der Daten zurückgibt.

# Nachteile des EM-Algorithmus



Was denken Sie?  
Wie könnten wir das in unserer  
Implementierung einbringen?



Was denken Sie?  
Was tun wir dagegen?

Das *globale Optimum* wird möglicherweise nicht gefunden

- Der EM-Algorithmus „verbessert sich“ zwar von Iteration zu Iteration, jedoch ist eine Konvergenz ins globale Optimum nicht garantiert  
→ Abhängigkeit von (u.a.) Initialbedingungen
- Daher startet man den Algorithmus häufig für unterschiedliche Startbedingungen
- und wählt dann das beste Ergebnis aus (oder aggregiert die Ergebnisse)
- sklearn führt diese mehrfache Initialisierung mit unterschiedlichen Anfangsbedingungen per default aus  
– einstellbar mittels des Eingabearguments `n_init`



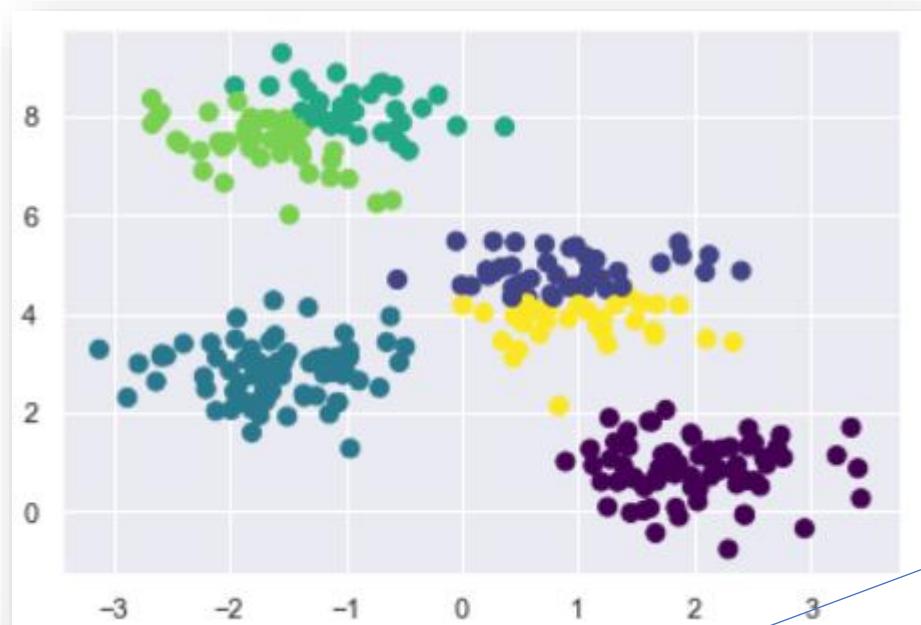
Was denken Sie?  
Was machen wir mit  
den Ergebnissen?

# Nachteile des EM-Algorithmus



Was denken Sie?

Was könnte man tun um eine geeignete Anzahl an Clustern zu bestimmen?



```
1 kmeans = KMeans(6, random_state=0)
2 labels = kmeans.fit_predict(X)
3 plt.scatter(X[:, 0], X[:, 1], c=labels,
4             s=50, cmap='viridis');
```

Die Anzahl an Clustern muss vorgegeben werden – der k-means lernt diese nicht eigenständig aus den Daten

- Hierbei handelt es sich um ein Problem, das viele *Unsupervised* Algorithmen betrifft
- Konkret können wir unserem k-means Modell aus sklearn eine beliebige Anzahl an zu extrahierenden Clustern vorgeben
- Man benötigt ein *Validierungskriterium*, um entscheiden zu können, welche Anzahl an extrahierten Clustern geeignet ist

# Validierung von Clustering



Was denken Sie?

Wie würden Sie (intuitiv) eine Internal Evaluation durchführen?  
Was könnte eine Heuristik sein?

”



”>”



Was denken Sie?

Wie würden Sie (intuitiv) bewerten,  
ob Ihr Clustering „gut“ war?

Verschiedene, grundlegende Möglichkeiten der Cluster-Validierung:

- **Internal Evaluation:** das Clustering wird durch ein Gütemaß repräsentiert und bewertet
- **External Evaluation:** das Clustering wird mit einer „Ground Truth“ an Labels verglichen
- **Manual Evaluation:** manuelle Evaluation durch einen Fachexperten
- **Indirect Evaluation:** Evaluation durch Anwendung des Clustering-Ergebnisses



So what?

Cluster-Validitätskriterien bewerten nach Heuristiken wie

- „inter-Cluster-Ähnlichkeit“ so gering wie möglich bzw. „- Distanz“ so groß wie möglich – und zugleich
- „intra-Cluster-Ähnlichkeit“ so hoch wie möglich bzw. „- Streuung“ so niedrig wie möglich

# Validierung von Clustering: Maße



## Dunn-Index:

- Intuitives Maß zur Bewertung eines Clusterings basierend auf dem Verhältnis aus der **Inter-Cluster-Distanz**  $\delta(C_i, C_j)$  zur **Intra-Cluster-Distanz**  $\Delta_i$

$$DI_m = \frac{\min_{1 \leq i < j \leq m} \delta(C_i, C_j)}{\max_{1 \leq k \leq m} \Delta_k}$$

wobei  $m$  der Anzahl an Clustern und  $C_i$  den Centroids entspricht

- Für die Inter-Cluster-Distanz könnte man die Abstände zwischen den Centroids definieren
- Für die Intra-Cluster-Distanz das Maximum der Abstände zwischen Datenpunkte innerhalb eines Clusters



Demo  
Gemeinsam am  
Whiteboard herleiten



Was denken Sie?

Können Sie sich das min  
und max erklären?

## Davies-Bouldin Index:

- Weniger intuitives Maß als der Dunn-Index
- Bewertet die Trennung zwischen den Clustern und die Streuung innerhalb der Cluster
- Er ist definiert als:

$$DB = \frac{1}{m} \sum_{i=1}^m \max_{i \neq j} \left( \frac{s_i + s_j}{d(C_i, C_j)} \right)$$

- Wobei  $s_i$  die Streuung der Abstände aller Datenpunkte zum jeweiligen Centroid innerhalb des Clusters  $i$  darstellt
- In `sklearn.metrics` gibt es eine Funktion hierfür  
`davies_bouldin_score(X, y)`

# Validierung von Clustering: Maße

## Silhouette Score: Beschreibung

- Bewertet, wie **ähnlich** ein Datenpunkt zu seinem **eigenen** und wie **unähnlich** zu den **übrigen Clustern** ist
- Er ist so definiert, dass sein **Wertebereich** zwischen -1 und 1 liegt
- Interpretation der Wertebereiche:
  - **Nahe 1:** Datenpunkt liegt in einem Cluster
  - **Um 0:** Datenpunkt liegt zwischen zwei Clustern
  - **< 0:** Datenpunkte eines nahegelegenen Clusters liegen näher am betrachteten Datenpunkt → Clustering kann verbessert werden
- Weisen also viele Datenpunkte einen **hohen Silhouette-Score** auf, dann liegt ein **geeignetes Clustering** vor
- Typisch sind auch Plots des Silhouette Scores

## Silhouette Score: Definition

- Wir benötigen den mittleren Abstand eines Datenpunktes zu allen anderen eines Clusters  $A$

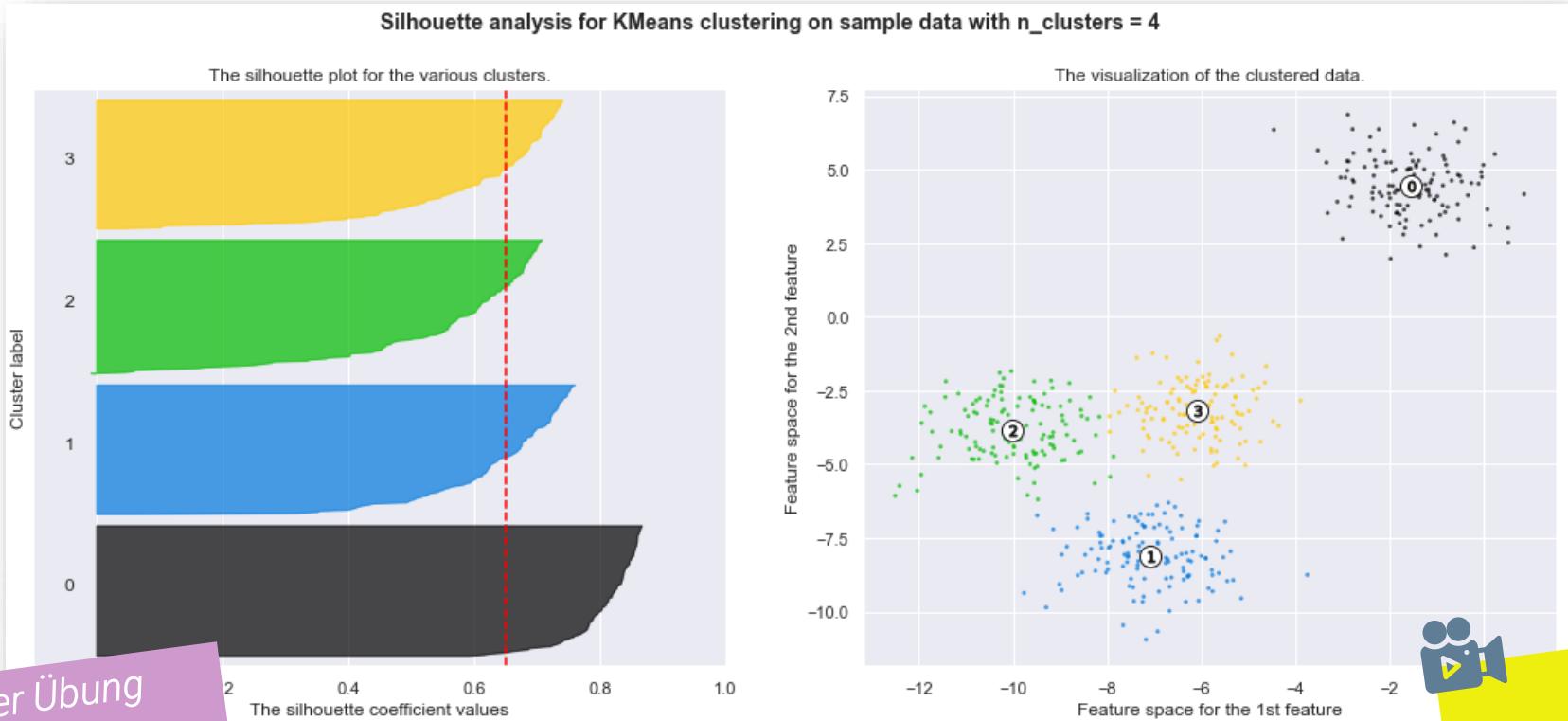
$$\text{dist}(A, o) = \frac{1}{n_A - 1} \sum_{a \in A, a \neq o} \text{dist}(a, o)$$

- Distanz zum nächstgelegenen Cluster  $B$ : den kleinsten, mittleren Abstand eines Datenpunktes zu allen anderen Datenpunkten der nicht-eigenen Cluster  $C$

$$\text{dist}(B, o) = \min_{C \neq A} \underbrace{\left( \frac{1}{n_C} \sum_{c \in C} \text{dist}(c, o) \right)}_{=\text{dist}(C, o)}$$

- Damit definieren wir den Silhouette Score für einen Datenpunkt
  - wenn  $o$  einziges Element von  $A$  ist
  - sonst
- Und der Mittelwert hiervon über alle Datenpunkte ist ein Gütemaß für einen Clustering-Vorgang

# Validierung von Clustering: Silhouette-Plot



Möglichkeit einer Übung  
Cluster-Validierung kann  
problematisch sein: sehen wir  
uns in einer Übungsaufgabe  
noch genauer an



Demo  
Ausführung im  
Jupyter Notebook

# Wie lesen und interpretieren wir einen Silhouette-Plot?



## Interpretationshilfen/-heuristiken

- Gibt es viele Cluster/Datenpunkte mit Silhouette-Scores **kleiner** als der **durchschnittliche Score**, so spricht das für ein schlechtes Clustering
- **Negative Silhouette-Scores** sind immer kritisch zu betrachten und deuten häufig auf ein schlechtes Clustering hin
- Allgemein: eine heuristische – aber nicht pauschal zutreffende – Annahme: **natürliche Cluster haben oft vergleichbare Größen**. Gibt es starke Unterschiede → evtl. schlechtes Clustering

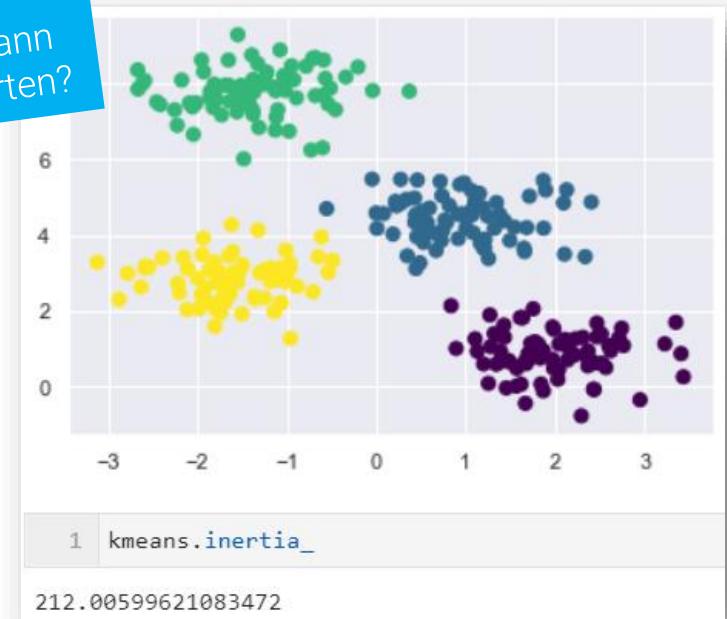
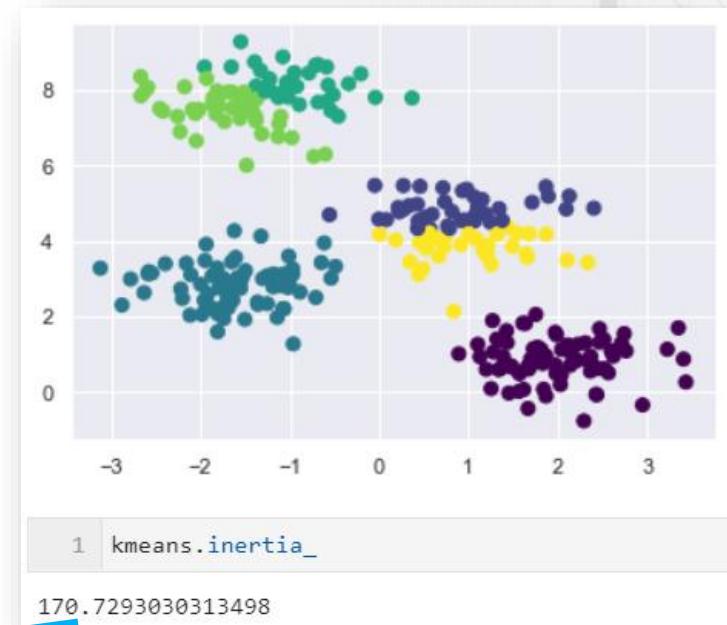
# Validierung von Clustering: Maße

## SSE und Elbow Plots:

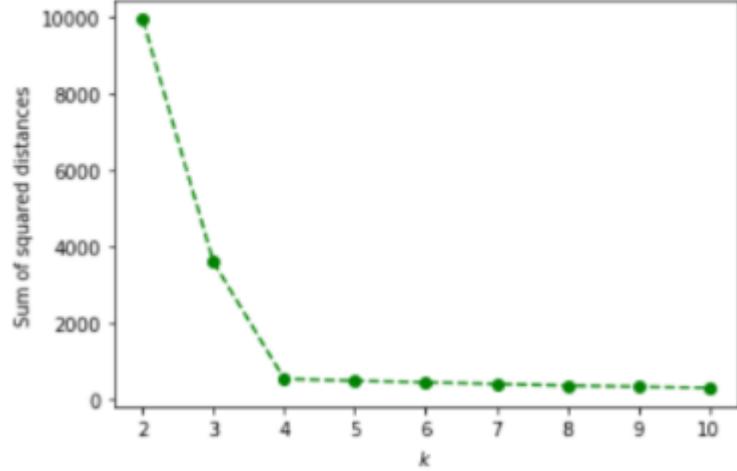
- Eine weitere, einfachere Möglichkeit zur Validierung bietet die Betrachtung der Sum of Squared Errors bzw. Sum of Squared Distances
- Hierzu berechnet man die Summe der quadratischen Abstände zwischen allen Datenpunkten und ihren jeweilig zugehörigen Centroids
- In `sklearn` liegt dieser Wert unter dem Attribut `.inertia_` des Modells
- Dieses Maß trägt man dann in einem sog. *Elbow Plot* bzw. *Knee Plot* bzw. *Scree Plot* auf
- Die optimale Anzahl an Clustern liegt vor, wenn dieser Graph einen „Knick macht“
- Grundgedanke: ab diesem Punkt erklären weitere Cluster zu wenig Varianz der Daten



Was denken Sie?  
Was tun wir dann  
mit diesen Werten?



# Beispiel: Cluster-Validierung mittels Elbow-Plot



In unserem selbst erzeugten Beispiel stecken vier Cluster. Wir wollen nun mittels der Elbow-Plot-Methode validieren, dass auch wirklich ein Clustering, das vier Cluster extrahiert, die Struktur der Daten am besten repräsentiert. Unsere Aufgabe ist also mittels eines 'for'-Loops k-means mit ansteigendem  $k$  an unseren Daten zu trainieren und die Sum of Squared Distanzen eines jeden Trainingsdurchgangs in einem Linienplot darzustellen.