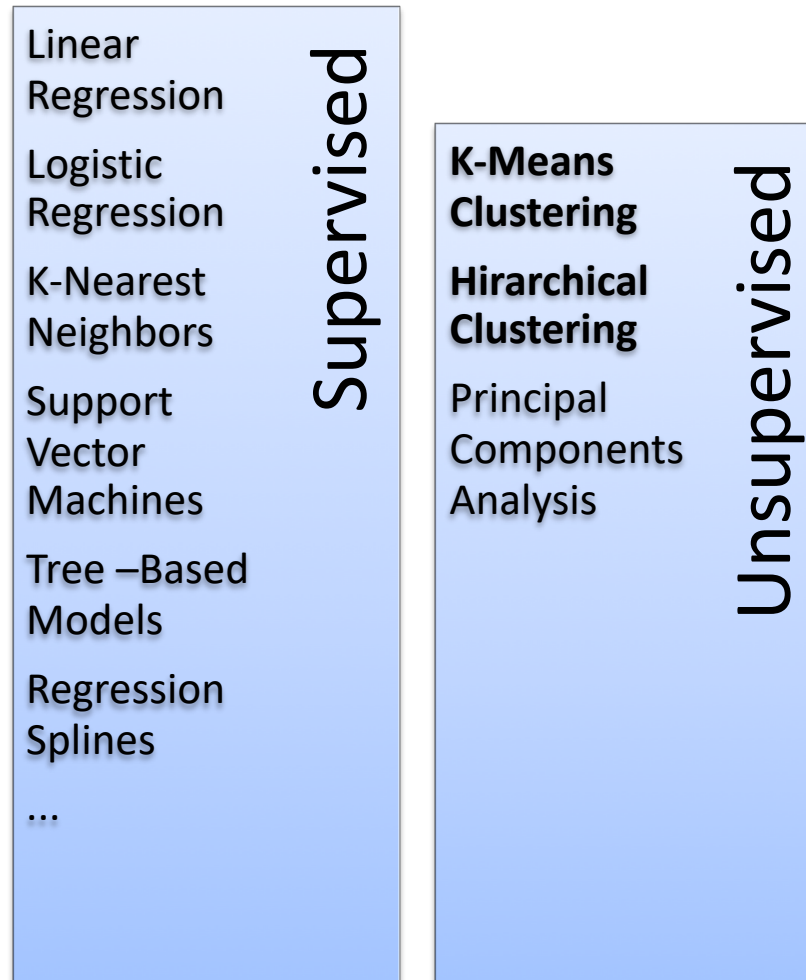
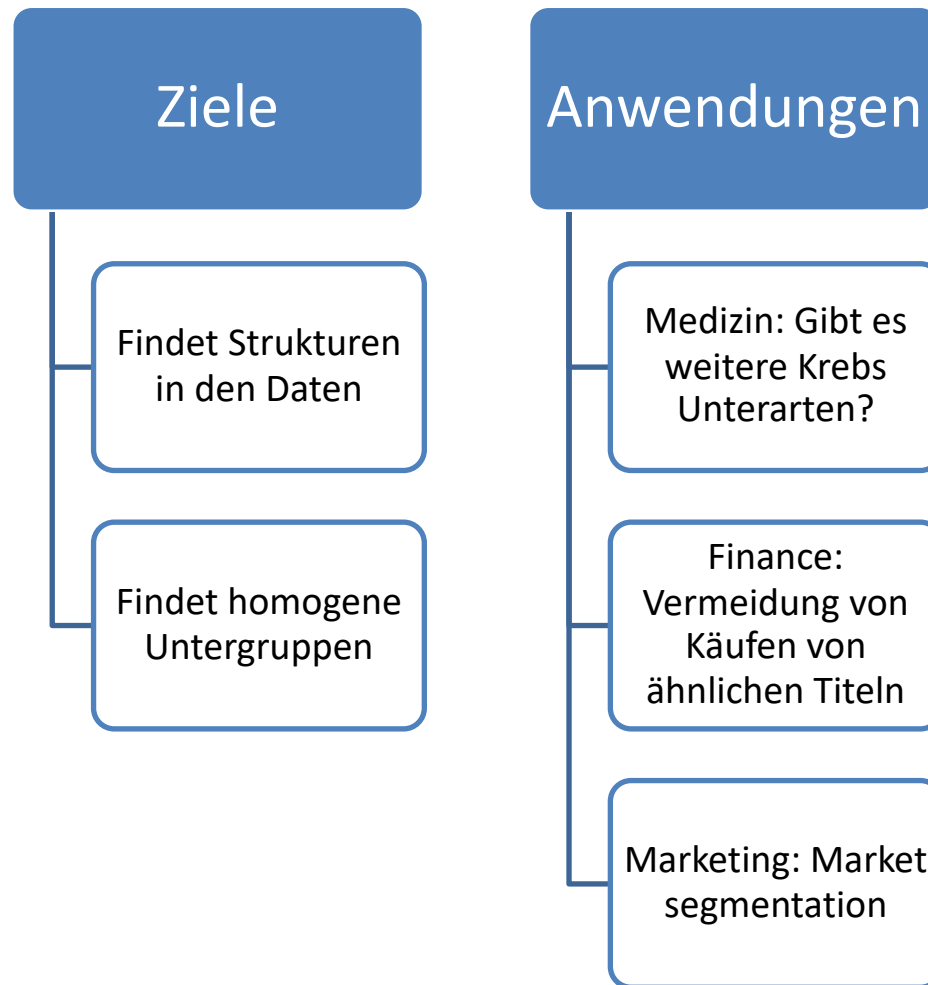


“Der Clustering Algorithmus”

Supervised vs. Unsupervised



Ziele des Clustering und Anwendungen



K-Means Clustering

1. Formale Eigenschaften

Datenset mit n
Beobachtungen und p
Ausprägungen

K Cluster (im Vorhinein
festgelegt)

Jede Beobachtung
gehört zu einem
Cluster:

$$C_1 \cup \dots \cup C_k = \{1 \dots n\}$$

Die Cluster sind nicht
überlappend:

$$C_k \cap C_{k'} = \emptyset \quad \forall k' \neq k$$

K-Means Clustering

2. Minimierung der Zielfunktion

$$\min_{C_1, \dots, C_k} \sum_{k=1}^K W(C_k)$$

mit

$$W(C_k) = \frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^p (x_{ij} - \overline{x_{kj}})^2$$

K-Means Clustering

3. Der Algorithmus

1.

- Ordne zufällig jede Beobachtung einem Cluster zu.

2.

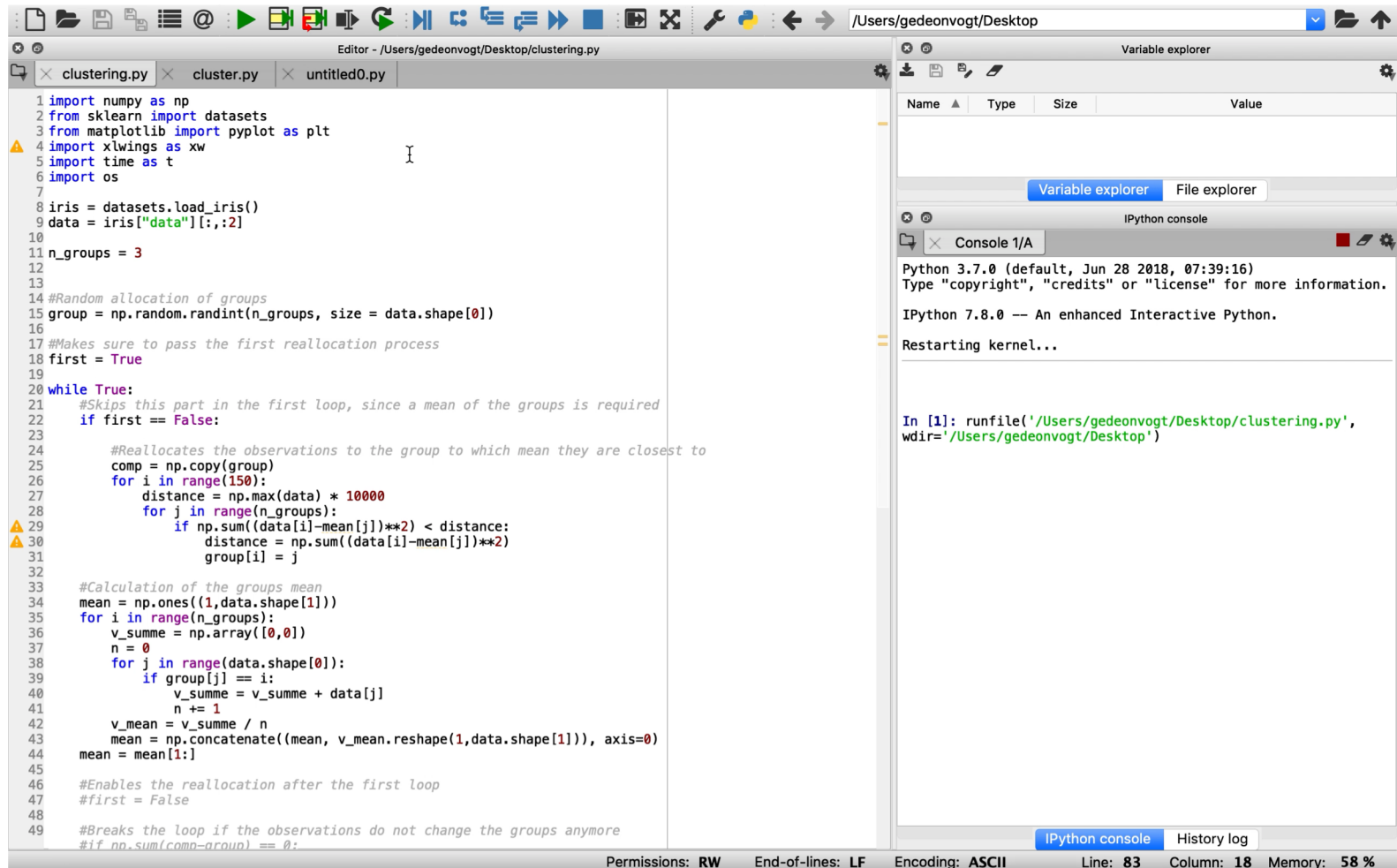
- a) Berechne den *Centroid* aller Cluster.
- b) Ordne jede Beobachtung dem Cluster zu, dessen *Centroid* am nächsten ist.

3.

- Wiederhole 2. so lange, bis sich die Zuordnung nicht mehr ändert.

K-Means Clustering

4. Das ganze in Aktion



The screenshot displays a Jupyter Notebook environment. The main window is a code editor with a file named `clustering.py`. The code implements a K-Means clustering algorithm using NumPy and SciPy. It loads the Iris dataset, initializes 3 clusters, and iteratively assigns points to the nearest cluster and recalculates the cluster means until convergence.

```
1 import numpy as np
2 from sklearn import datasets
3 from matplotlib import pyplot as plt
4 import xlwings as xw
5 import time as t
6 import os
7
8 iris = datasets.load_iris()
9 data = iris["data"][:, :2]
10
11 n_groups = 3
12
13
14 #Random allocation of groups
15 group = np.random.randint(n_groups, size = data.shape[0])
16
17 #Makes sure to pass the first reallocation process
18 first = True
19
20 while True:
21     #Skips this part in the first loop, since a mean of the groups is required
22     if first == False:
23
24         #Reallocates the observations to the group to which mean they are closest to
25         comp = np.copy(group)
26         for i in range(150):
27             distance = np.max(data) * 10000
28             for j in range(n_groups):
29                 if np.sum((data[i]-mean[j])**2) < distance:
30                     distance = np.sum((data[i]-mean[j])**2)
31                     group[i] = j
32
33         #Calculation of the groups mean
34         mean = np.ones((1,data.shape[1]))
35         for i in range(n_groups):
36             v_summe = np.array([0,0])
37             n = 0
38             for j in range(data.shape[0]):
39                 if group[j] == i:
40                     v_summe = v_summe + data[j]
41                     n += 1
42             v_mean = v_summe / n
43             mean = np.concatenate((mean, v_mean.reshape(1,data.shape[1])), axis=0)
44         mean = mean[1:]
45
46         #Enables the reallocation after the first loop
47         #first = False
48
49         #Breaks the loop if the observations do not change the groups anymore
50         #if np.sum(comp-group) == 0:
```

On the right side, there is a 'Variable explorer' panel showing a table with columns 'Name', 'Type', 'Size', and 'Value'. Below it is an 'IPython console' panel showing the command `runfile('/Users/geideonvogt/Desktop/clustering.py', wdir='/Users/geideonvogt/Desktop')` and the output `Restarting kernel...`. At the bottom of the console, it shows `In [1]:` followed by the same `runfile` command.

At the bottom of the Jupyter interface, there is a status bar with the following information: Permissions: RW, End-of-lines: LF, Encoding: ASCII, Line: 83, Column: 18, Memory: 58 %.

K-Means Clustering

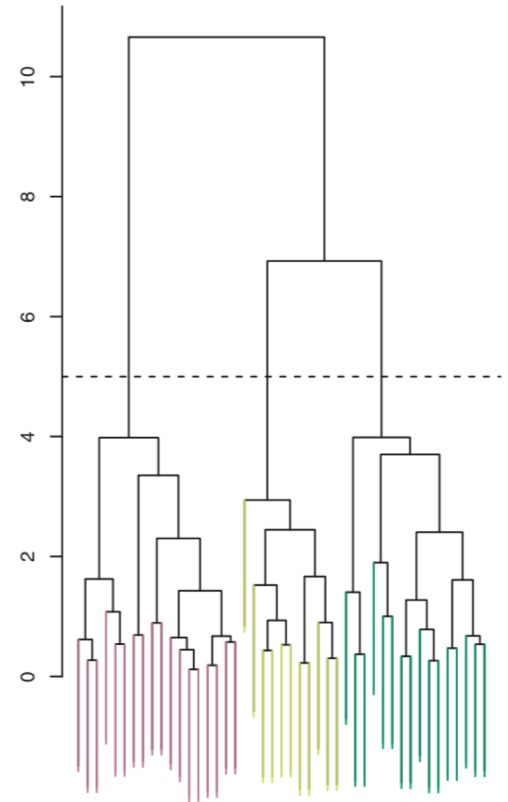
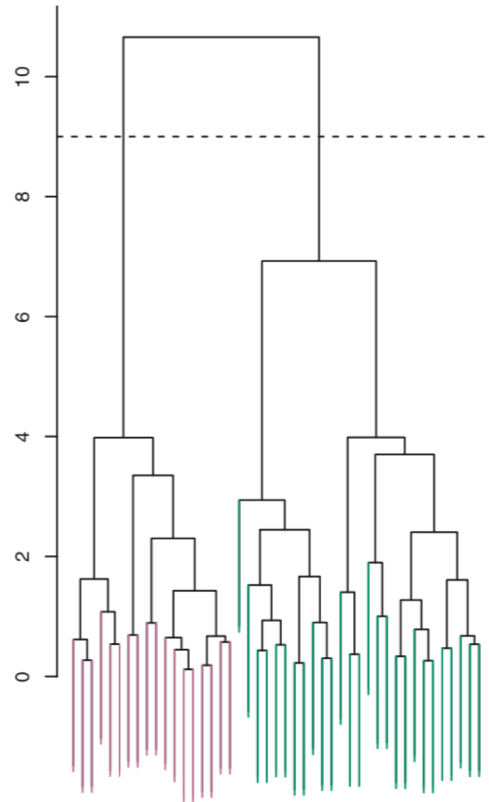
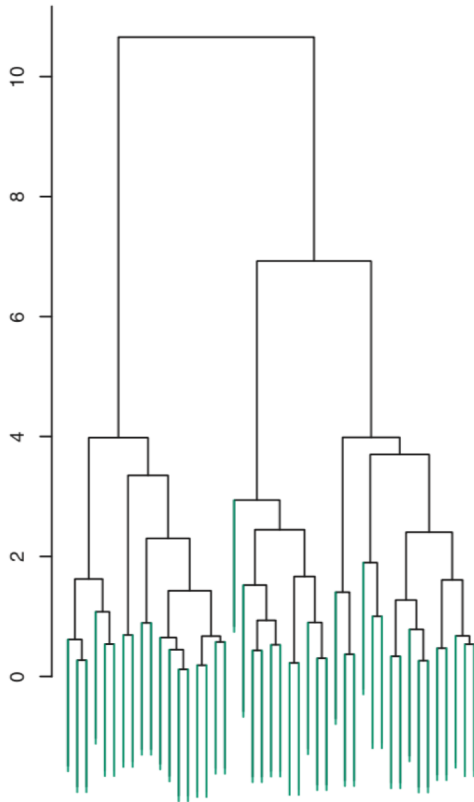
5. Probleme

Der Algorithmus findet nur *lokale* Optima. *Globale* Optima hängen von der zufälligen Zuordnung der Beobachtungen am Anfang ab.

Wiederhole den Algorithmus mehrere Male mit unterschiedlichen Startzuordnungen und wähle zum Schluss das Ergebnis mit der geringsten durchschnittlichen Distanz.

Hierarchical Clustering

1. Dendrogram



Hierarchical Clustering

2. Der Algorithmus

1.

- Beginne damit, alle paarweise "dissimilarities" (z.B. Euklidische Distanz) zu bestimmen. Behandle jede Beobachtung als ein eigenes Cluster.

2.

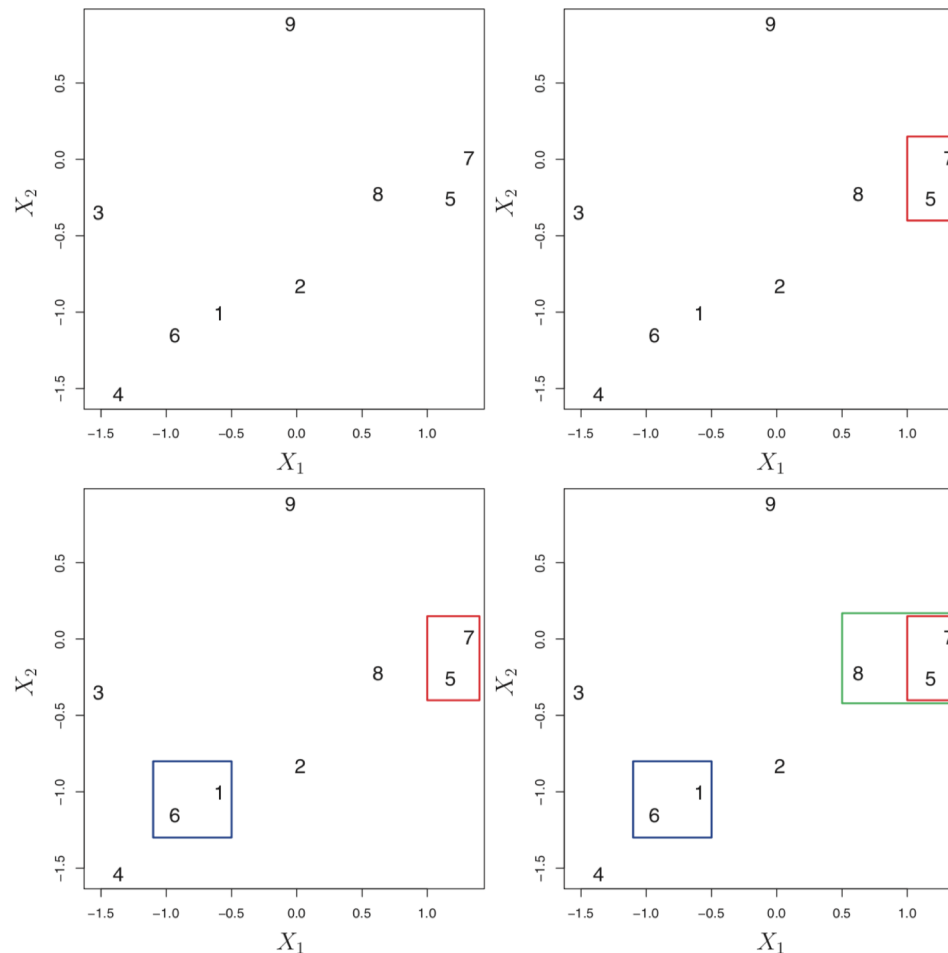
- Bestimme alle "dissimilarities" zwischen den Clustern und kombiniere die beiden Cluster, die am ähnlichsten sind, zu einem neuen. Die "dissimilarities" bestimmen die Höhe im Dendrogramm.
- Berechne nun erneut die "dissimilarities" zwischen die Clustern für verblieben $i-1$.

3.

- Wiederhole 2. für $i = n, n-1, \dots, 2$

Hierarchical Clustering

3. Grafische Darstellung des Ablaufs



Wahl von “dissimilarity” Maßen

"Dissimilarity" = Der Abstand zwischen zwei Punkten/Beobachtungen

Euklidischer Abstand

Korrelationsbasierter Abstand

Manhattan Abstand

Sollten wir die Variablen zusätzlich noch Standardisieren?

Verschiedene Arten von “linkage”

Complete Linkage

- Berechne alle Abstände zwischen den Beobachtungen die in Cluster A enthalten sind und der zuzuordnenden Beobachtung B. Die Complete Linkage ist der größte Abstand.

Single Linkage

- Der geringste Abstand zwischen allen Beobachtungen aus Cluster A und Beobachtung B.

Average Linkage

- Durchschnitt aller Abstände zwischen allen Beobachtungen aus Cluster A und Beobachtung B.

Centroid

- Abstand zwischen Beobachtung B und dem Durschnitt aus allen Beobachtungen, die in Cluster A enthalten sind.

Was soll man wählen?

- Average und Complete Linkage werden generell Single Linkage gegenüber bevorzugt, da sie tendenziell eher zu ausgeglichenen Dendrogrammen führen.
- Centroid sollte eher mit Bedacht genutzt werden, da dies zu „inversions“ führen kann.

Welche Entscheidungen müssen wir treffen?

- Sollten die Daten Standardisiert werden? (Zero mean; standard deviation von 1)
- Beim Hierarchical Clustering:
 - Welches Maß für die „dissimilarities“ sollen wir wählen?
 - Welche Art „linkage“ sollen wir nehmen?
 - Wo sollen wir das Dendrogramm abschneiden?
- Beim K-Means Clustering:
 - Wie viele Cluster vermuten wir in den Daten?