

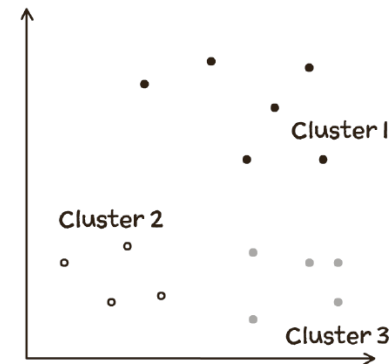
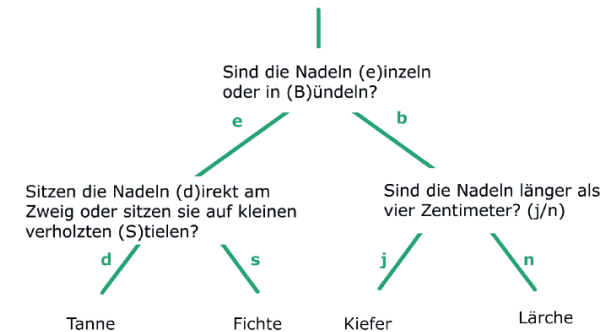
Künstliche Intelligenz kapiern und programmieren

Teil 3: Klassifizieren

Michael Weigend
Universität Münster



mw@creative-informatics.de
www.creative-informatics.de
2024



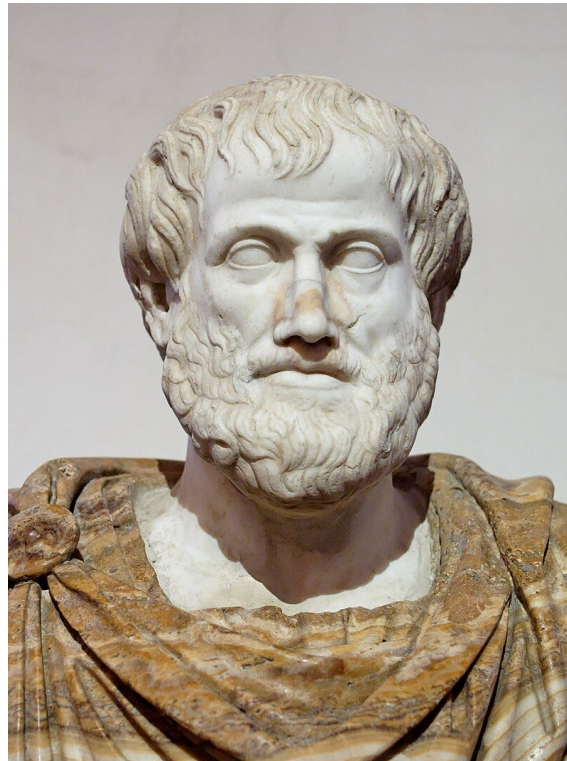
Materialien bei GitHub:
<https://github.com/mweigend/ki-workshop>

Tag 1

| Zeit | Thema | Inhalte |
|-------|---------------------|--|
| 9.00 | Denkende Maschinen | Pädagogische Konzepte, Einstieg in Python, Chatbots und Assistenzsysteme |
| 11.15 | KI als Spielgegner | Modellieren mit Listen, Nim-Spiel mit KI als Gegner |
| 12.30 | <i>Mittagspause</i> | |
| 13.30 | Klassifizieren | Entscheidungsbaum, k-Means-Clustering |
| 14.45 | Lernen | Lernfähiger Währungsrechner, Wegsuche, Fußgänger erkennen |
| 16.00 | <i>Ende</i> | |

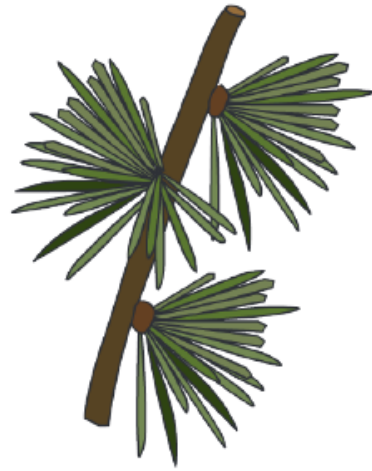
Wie könnte man uns in Gruppen einteilen?

Man kann die Dinge der Welt mit Hilfe von Merkmalen klassifizieren

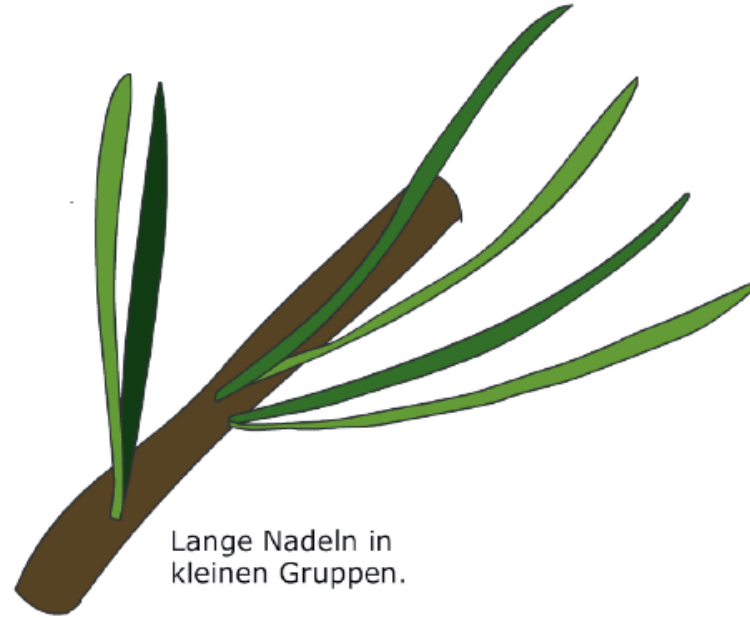


Aristoteles

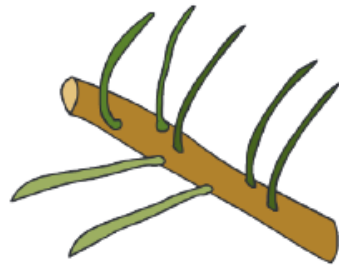
Nadelbäume bestimmen



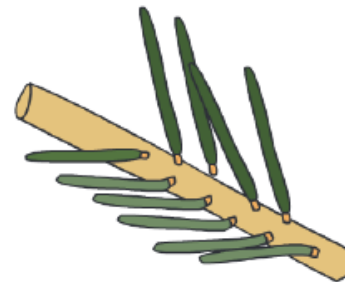
Bündel aus vielen Nadeln.



Lange Nadeln in kleinen Gruppen.



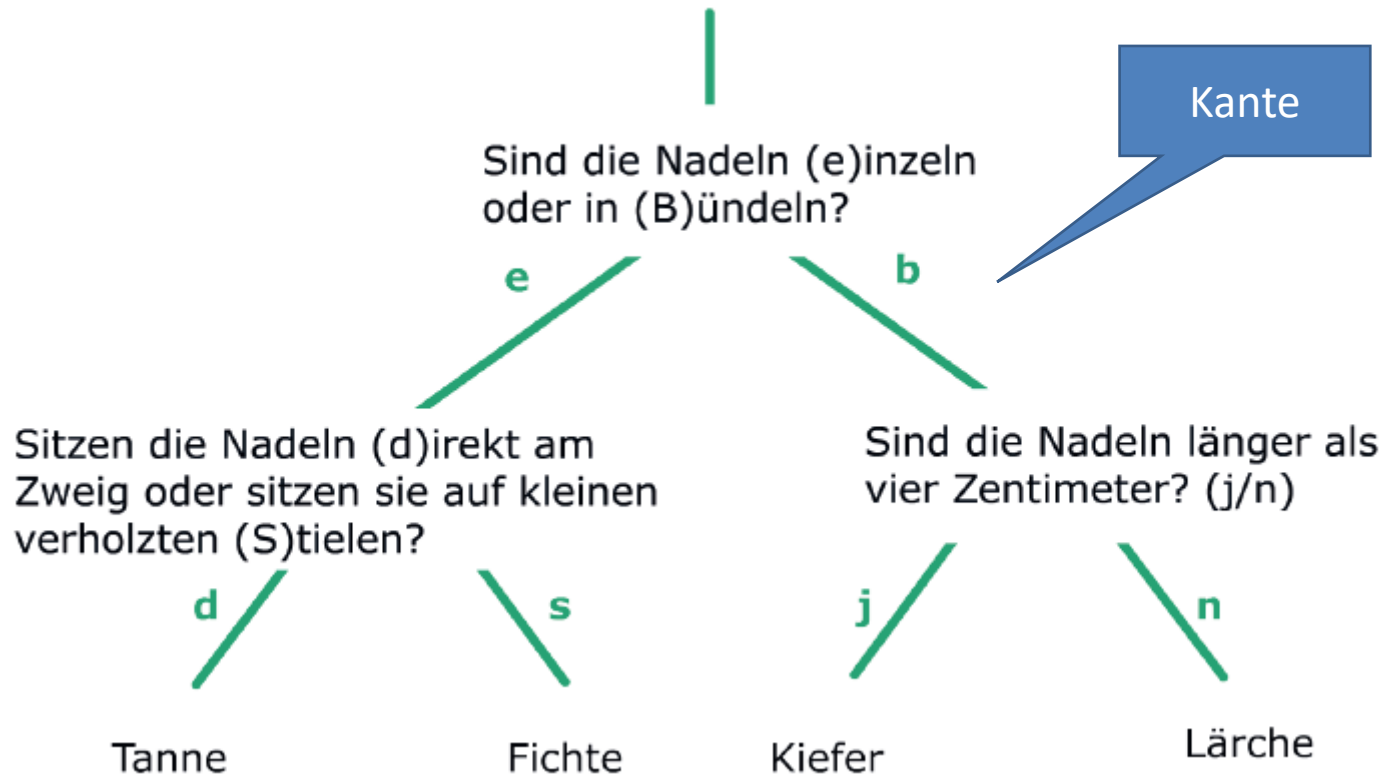
Einzelne Nadeln, die direkt auf dem Zweig sitzen.



Einzelne Nadeln, die auf kleinen verholzten Stielen sitzen.

Nadelbäume bestimmen

Entscheidungsbaum

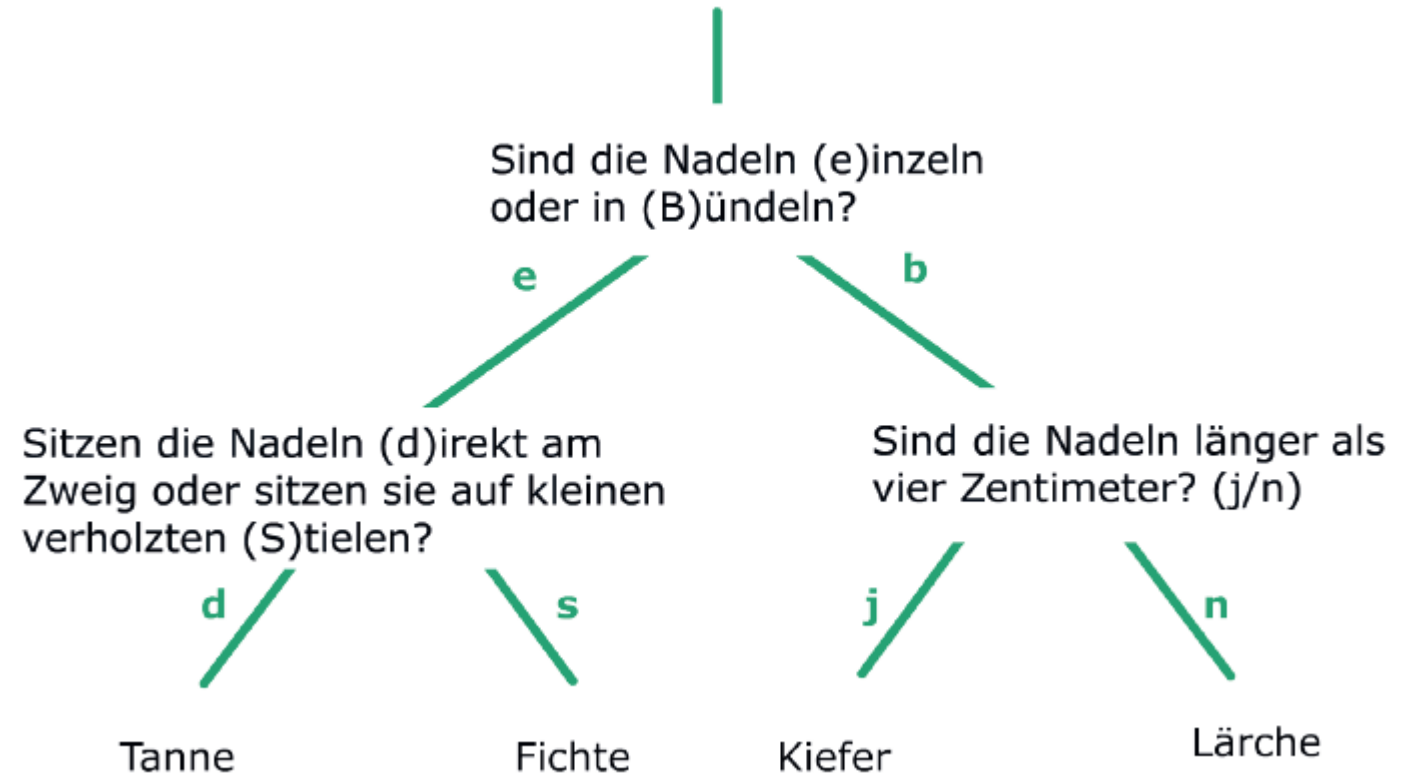


Kante

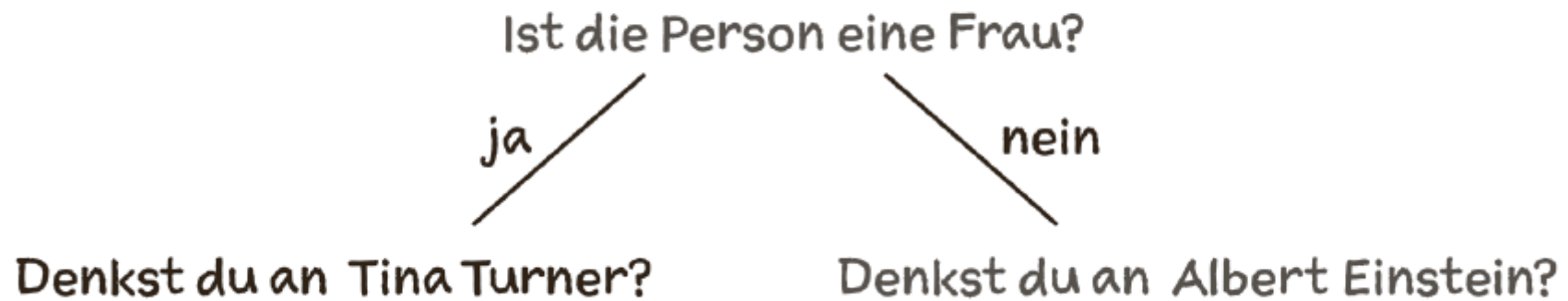
Knoten

Blatt

Von welchem Baum stammt dieser Zweig?



Prominente raten



Projekt: An wen denkst du?

Ein lernfähiges Programm

Der Spieler denkt an eine prominente Persönlichkeit. Der Computer ermittelt durch Ja/Nein-Fragen den Namen der Persönlichkeit. Wenn die Vorhersage nicht zutrifft, fragt der Computer nach dem Namen der Persönlichkeit und nach einem Unterscheidungsmerkmal. Dann erweitert er seinen Entscheidungsbaum.

Die Software lernt dazu und wird immer besser.

Beispieldialog

Denke an eine berühmte Person.

Ist die Person eine Frau? (j/n): n

Denkst du an Albert Einstein? (j/n): n

Wie heißt die Person, an die du denkst?

Name: John Lennon

Denke dir eine Frage aus, mit der man die Person von der Person unterscheiden kann, die ich gerade genannt habe. Diese Frage muss man bei deiner Person mit "ja" und bei der anderen Person mit "nein" beantworten können.

Frage: Ist die Person ein Musiker?

Ist die Person ein Musiker? (j/n): j

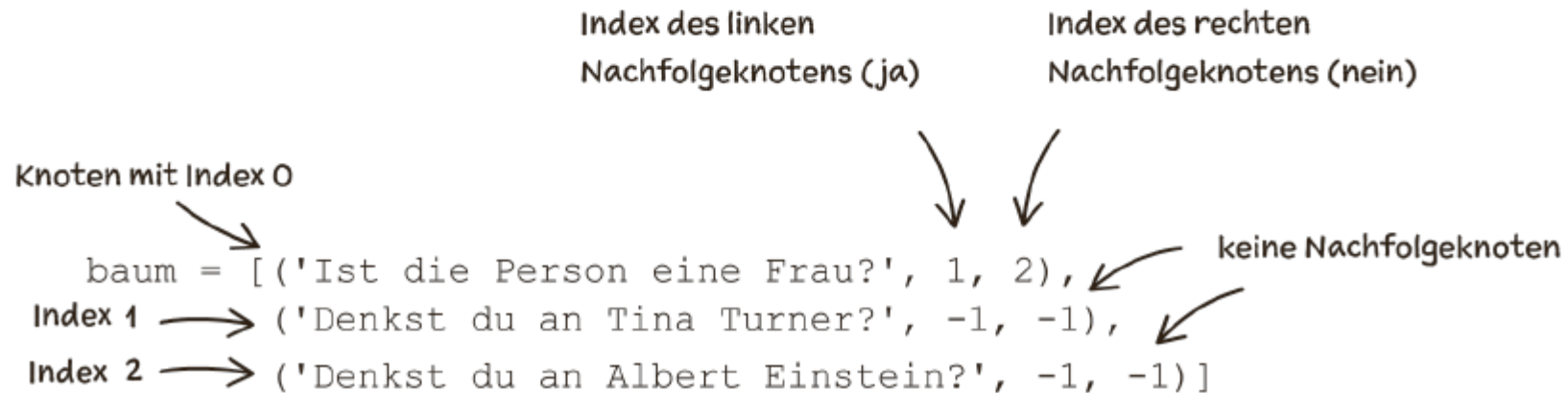
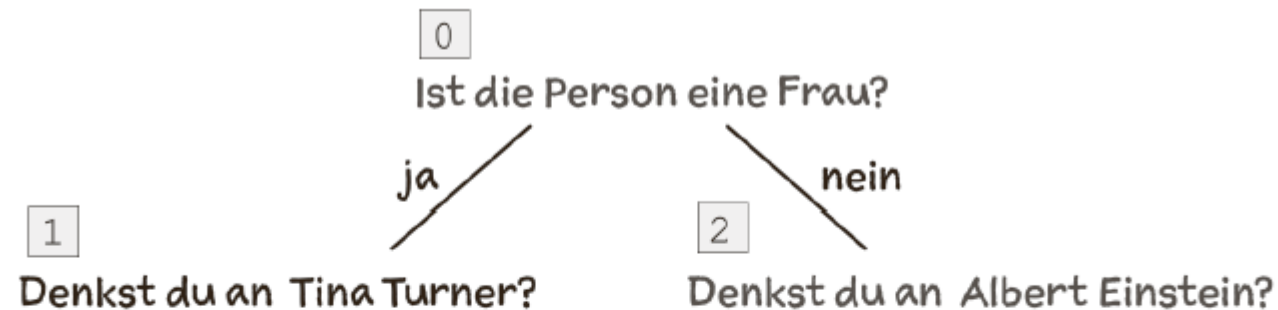
Denkst du an John Lennon? (j/n): j

Die Person wurde gefunden.

Welche Aufgaben muss das Programm lösen?

- Den Entscheidungsbaum implementieren.
- Den Entscheidungsbaum laden und speichern.
- Fragen stellen, bis ein Blatt des Entscheidungsbaums erreicht ist.
- Entscheidungsbaum erweitern.

Ein gerichteter Baum als Liste von Tupeln



Teil 1: Vorbereitung

wird
versuchsweise
ausgeführt

```
import pickle
```

Dateiname

```
try:
```

```
    stream = open('daten.dat', mode='rb')  
    baum = pickle.load(stream)  
    stream.close()
```

Binärdatei zum Lesen öffnen

Aus der Binärdatei wird
die Datenstruktur des
Baums gewonnen

```
except:
```

```
    baum = [('Ist die Person eine Frau?', 1, 2),  
            ('Denkst du an Tina Turner?', -1, -1),  
            ('Denkst du an Albert Einstein?', -1, -1)]
```

```
    aktuell = 0
```

```
    gefunden = False
```

```
    print('Denke an eine berühmte Person.')
```

wird ausgeführt,
falls der Versuch
scheitert

Teil 2:

Interaktion

Das Tupel wird ausgepackt

```
while not gefunden:
    frage, ja, nein = baum[aktuell]
    eingabe = input(frage + ' (j/n): ')
    if ja == -1:
        if eingabe == 'j':
            gefunden = True
        else:
            print('Wie heißt die Person, an die du denkst?')
            name = input('Name: ')
            print('Denke dir eine Frage aus, mit der man ')
            print('die Person von der Person unterscheiden kann,')
            print('die ich gerade genannt habe. Diese Frage')
            print('muss man bei deiner Person mit "ja" und bei der')
            print('anderen Person mit "nein" beantworten können.')
            neueFrage = input('Frage: ')
            jaNeu = len(baum)
            neinNeu = jaNeu + 1
            baum[aktuell] = (neueFrage, jaNeu, neinNeu)
            baum += [('Denkst du an ' + name + '?', -1, -1)]
            baum += [(frage, -1, -1)]
    else:
        if eingabe == 'j':
            aktuell = ja
        else:
            aktuell = nein
```

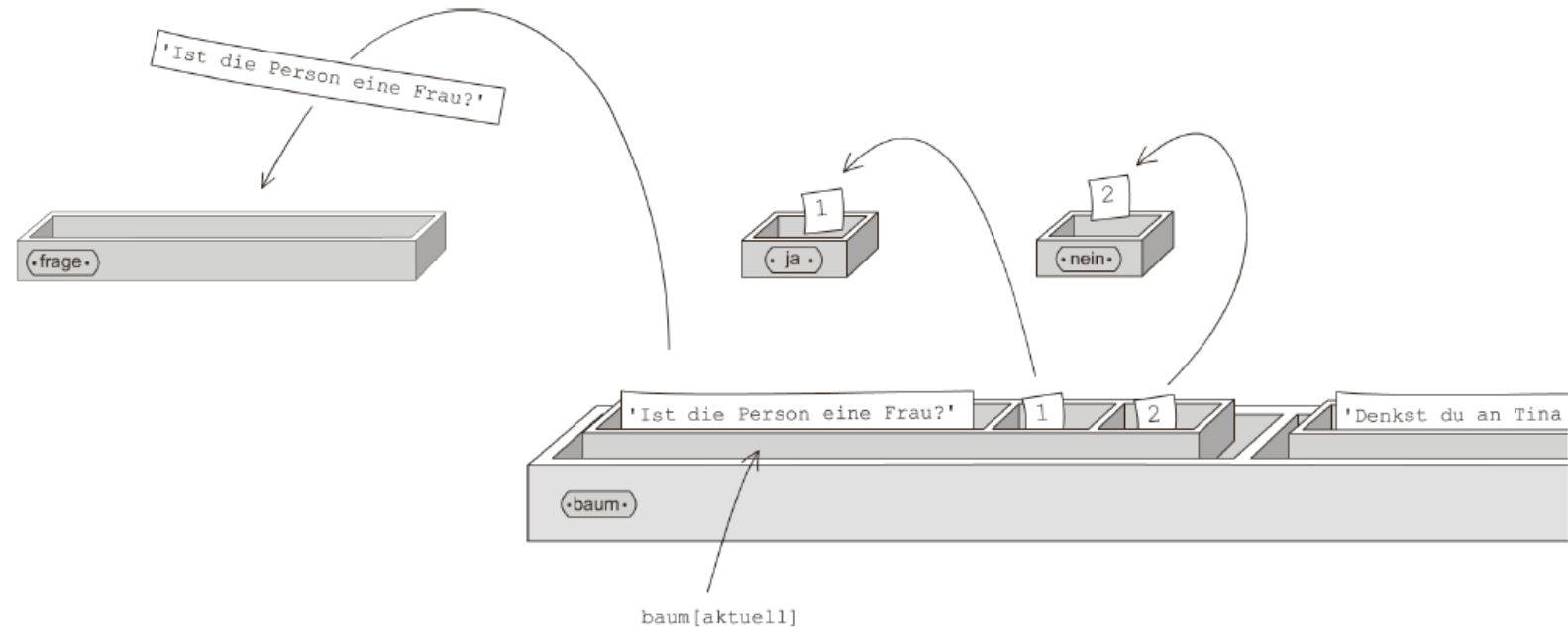
Neue Unterscheidungsfrage

Neue Personfrage (ja)

Alte Personfrage (nein)

Auspacken

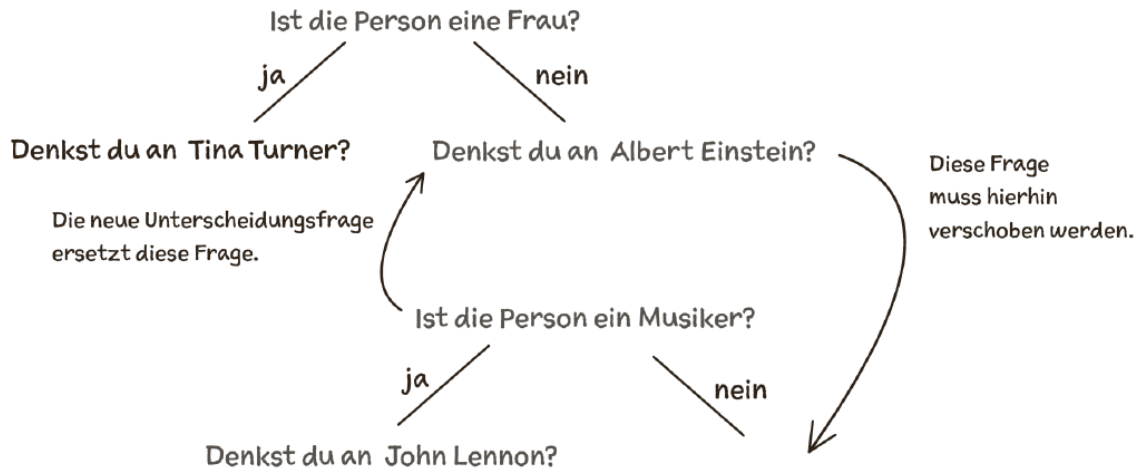
```
while not gefunden:  
    frage, ja, nein = baum[aktuell]
```



Teil 2:

Interaktion

Beispiel: Spieler denkt
an John Lennon



```
[('Ist die Person eine Frau?', 1, 2),  
( 'Denkst du an Tina Turner?', -1, -1),  
( 'Denkst du an Albert Einstein?', -1, -1)]
```

```
while not gefunden:  
    frage, ja, nein = baum[aktuell]  
    eingabe = input(frage + ' (j/n): ')  
    if ja == -1:  
        if eingabe == 'j':  
            gefunden = True  
        else:  
            print('Wie heißt die Person, an die du denkst?')  
            name = input('Name: ')  
            print('Denke dir eine Frage aus, mit der man ')  
            print('die Person von der Person unterscheiden kann,')  
            print('die ich gerade genannt habe. Diese Frage')  
            print('muss man bei deiner Person mit "ja" und bei der')  
            print('anderen Person mit "nein" beantworten können.')  
            neueFrage = input('Frage: ')  
            jaNeu = len(baum)  
            neinNeu = jaNeu + 1  
            baum[aktuell] = (neueFrage, jaNeu, neinNeu)  
            baum += [('Denkst du an ' + name + '?', -1, -1)]  
            baum += [(frage, -1, -1)]  
    else:  
        if eingabe == 'j':  
            aktuell = ja  
        else:  
            aktuell = nein
```


Teil 3: Schluss

```
print('Die Person wurde gefunden.')  
stream = open('daten.dat', mode='wb')  
pickle.dump(baum, stream)  
stream.close()  
input()
```

Binärdatei zum
Schreiben öffnen

Entscheidungsbaum in
Datei speichern

Warte bis ENTER
gedrückt

Übung 3.1 Ein lernfähiges Programm

- 1) Machen Sie aus dem Starterprojekt ein Ratespiel zu einem neuen Thema (z.B. Märchenfiguren, Automarken, Filmschauspieler, Säugetiere, evtl. fachübergreifendes Projekt ...)
- 2) Verbessern Sie die technische Qualität des Programmtextes, indem Sie Funktionen definieren.
- 3) Verändern Sie das Programm, so dass seine Arbeitsweise transparent wird. Z.B. könnte an bestimmtem Stellen des Programmlaufs die Anzahl der gespeicherten Prominenten oder der komplette Entscheidungsbaum ausgegeben werden.
- 4) (*) Machen Sie das Programm objektorientiert und definieren Sie eine Klasse für den Entscheidungsbaum.

Bitte bearbeiten Sie nur eine dieser Aufgaben.

Weitere Ideen?

https://docs.google.com/document/d/140INslEWA_AwUwtpVMCZqtH7_eOkU8rmvfrgWqE5M3E/edit?usp=sharing

Lösungen 3.1

Aufgabe 2

Definition einer Funktion

```
import pickle

def ladeBaum():
    try:
        stream = open('daten.dat', mode='rb')
        baum = pickle.load(stream)
        stream.close()
    except:
        baum = [('Ist die Person eine Frau?', 1, 2),
                ('Denkst du an Tina Turner?', -1, -1),
                ('Denkst du an Albert Einstein?', -1, -1)]
    return baum

def erweitereBaum(baum, name, neueFrage):
    jaNeu = len(baum)
    neinNeu = jaNeu + 1
    baum[aktuell] = (neueFrage, jaNeu, neinNeu)
    baum += [('Denkst du an ' + name + '?', -1, -1)]
    baum += [(frage, -1, -1)]

def speichereBaum(baum):
    stream = open('daten.dat', mode='wb')
    pickle.dump(baum, stream)
    stream.close()
```

Aufruf einer Funktion

```
# Hauptprogramm
aktuell = 0
gefunden = False
baum = ladeBaum()
print('Denke an eine berühmte Person.')
while not gefunden:
    frage, ja, nein = baum[aktuell]
    eingabe = input(frage + ' (j/n): ')
    if ja == -1:
        if eingabe == 'j':
            gefunden = True
        else:
            print('Wie heißt die Person, an die du denkst?')
            name = input('Name: ')
            print('Denke dir eine Frage aus, mit der man ')
            print('die Person von der Person unterscheiden kann, ')
            print('die ich gerade genannt habe. Diese Frage')
            print('muss man bei deiner Person mit "ja" und bei der')
            print('anderen Person mit "nein" beantworten können.')
            neueFrage = input('Frage: ')
            erweitereBaum(baum, name, neueFrage)
    else:
        if eingabe == 'j':
            aktuell = ja
        else:
            aktuell = nein

speichereBaum(baum)
print('Die Person wurde gefunden.')
input()
```

Aufgabe 3

```
import pickle

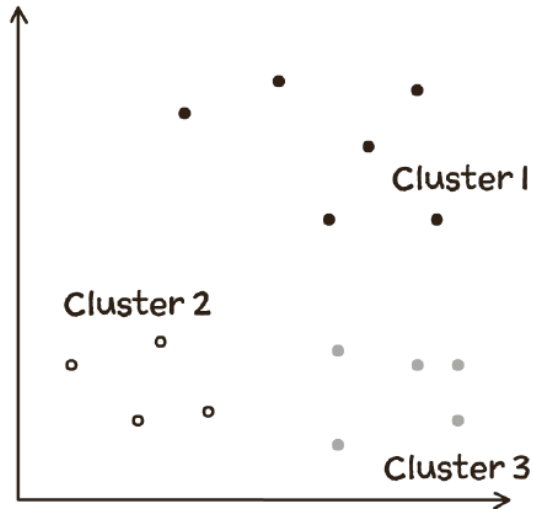
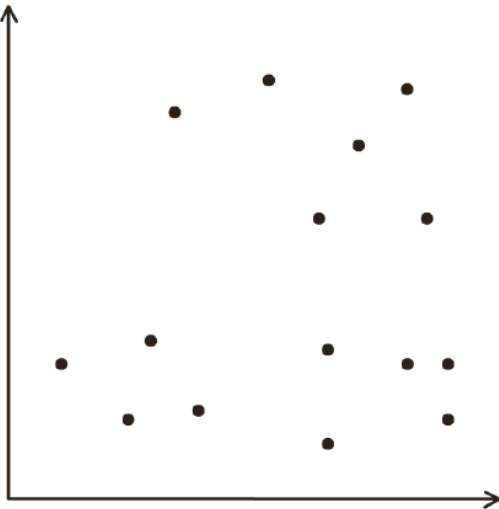
try:
    stream = open('daten.dat', mode='rb')
    baum = pickle.load(stream)
    stream.close()
except:
    baum = [('Ist die Person eine Frau?', 1, 2),
            ('Denkst du an Tina Turner?', -1, -1),
            ('Denkst du an Albert Einstein?', -1, -1)]

aktuell = 0
gefunden = False
print('Größe des Entscheidungsbaums:', len(baum))

...

print('Die Person wurde gefunden.')
for tupel in baum:
    print(tupel)
stream = open('daten.dat', mode='wb')
pickle.dump(baum, stream)
stream.close()
input()
```

Klassifizieren durch Clustern

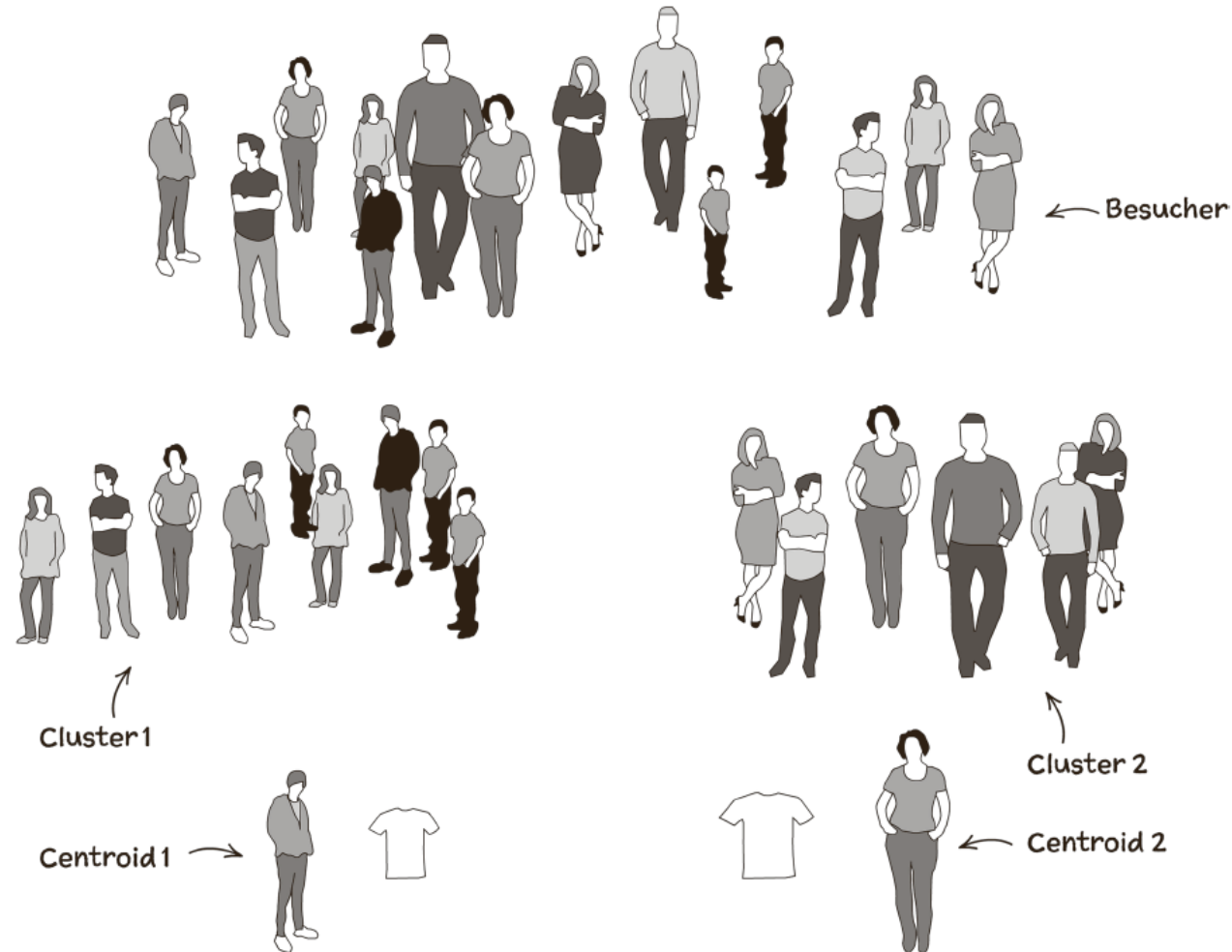


Ein *Cluster* besteht aus Datenpunkten, die näher zueinander als zu Datenpunkten außerhalb des Clusters stehen.

k-Means-Clustern: Die Datenpunkte werden in k Cluster aufgeteilt.

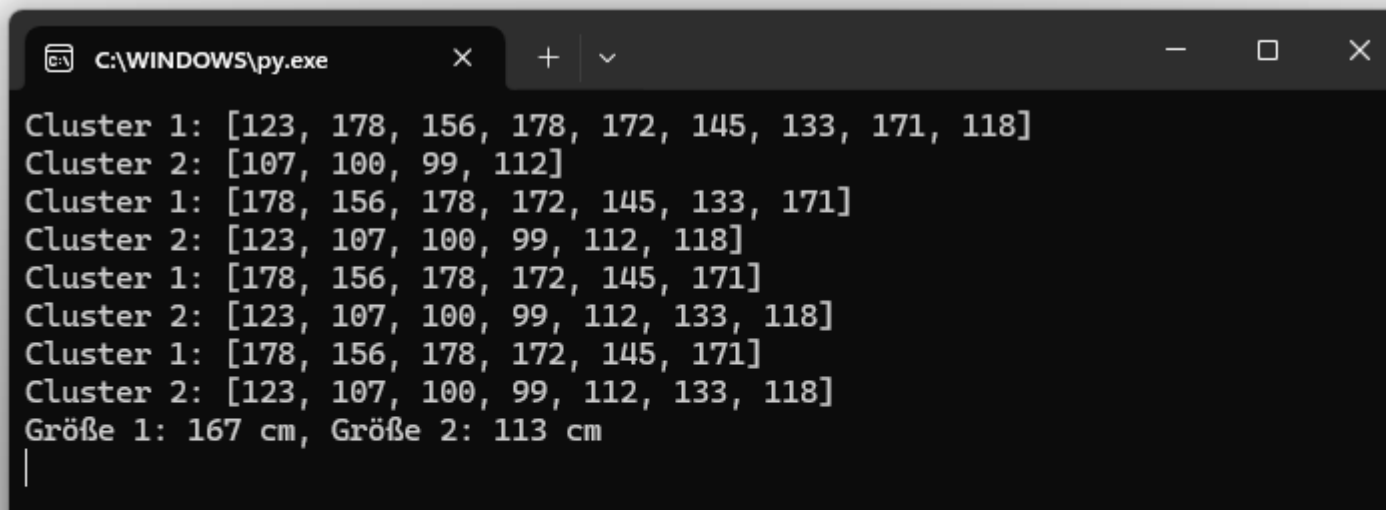
Projekt: T-Shirts für den Freizeitpark

$k = 2$



k-Means-Algorithmus (für $k = 2$)

1. Zu Beginn werden zwei beliebige Werte $c1$ und $c2$ aus der Liste der Körpergrößen als Centroiden ausgewählt.
2. Die Liste der Körpergrößen wird durchlaufen. Alle Werte, die näher an $c1$ liegen, werden dem ersten Cluster zugewiesen und die anderen Werte werden dem zweiten Cluster zugewiesen.
3. Jetzt werden die beiden Centroiden neu berechnet. Der Mittelwert des ersten Clusters ist $c1$ und der Mittelwert des zweiten Clusters ist $c2$.
4. Die Schritte 2 und 3 werden so lange wiederholt, bis sich nichts mehr ändert.



```
C:\WINDOWS\py.exe
Cluster 1: [123, 178, 156, 178, 172, 145, 133, 171, 118]
Cluster 2: [107, 100, 99, 112]
Cluster 1: [178, 156, 178, 172, 145, 133, 171]
Cluster 2: [123, 107, 100, 99, 112, 118]
Cluster 1: [178, 156, 178, 172, 145, 171]
Cluster 2: [123, 107, 100, 99, 112, 133, 118]
Cluster 1: [178, 156, 178, 172, 145, 171]
Cluster 2: [123, 107, 100, 99, 112, 133, 118]
Größe 1: 167 cm, Größe 2: 113 cm
```



```

# kmeans.py
körpergrößen = [123, 107, 178, 100, 99, 156, 178, 172, 112, 145, 133, 171, 118]
c1 = körpergrößen[0]
c2 = körpergrößen[1]
cluster1 = []
fertig = False

while not fertig:
    altesCluster1 = cluster1
    cluster1 = []
    cluster2 = []
    for g in körpergrößen:
        if (g-c1)**2 < (g-c2)**2:
            cluster1 += [g]
        else:
            cluster2 += [g]
    c1 = sum(cluster1)/len(cluster1)
    c2 = sum(cluster2)/len(cluster2)
    print('Cluster 1:', cluster1)
    print('Cluster 2:', cluster2)
    fertig = cluster1 == altesCluster1

print('Größe 1:', round(c1), 'cm,',
      'Größe 2:', round(c2), 'cm')
input()

```

Übung 3.2 Der k-Means-Algorithmus

1. Testen sie das Starterprojekt `kmeans.py`. Ändern Sie die Werte der Liste `körpergrößen` und beobachten Sie die Änderung der Ausgabe.
2. Entdecken Sie im Programmtext die Schritte des umgangssprachlichen Algorithmus. Fügen Sie Kommentare ein, die die Arbeitsweise des Programms erklären.
3. Ändern Sie das Programm ab, so dass es 3 Cluster bildet.
4. Ändern Sie das Programm ab, so dass es 1, ..., k Cluster bildet.

Weitere Ideen?

https://docs.google.com/document/d/140INslEWA_AwUwtpVMCZqtH7_eOkU8rmvfrgWqE5M3E/edit?usp=sharing

Lösungen 3.2

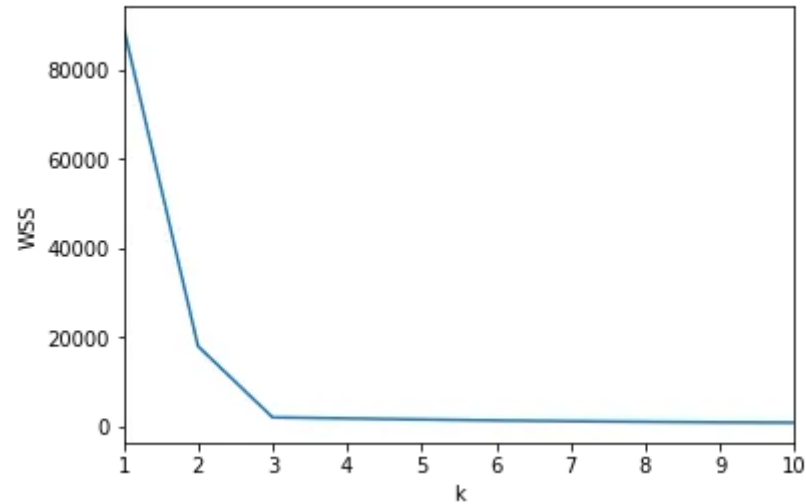
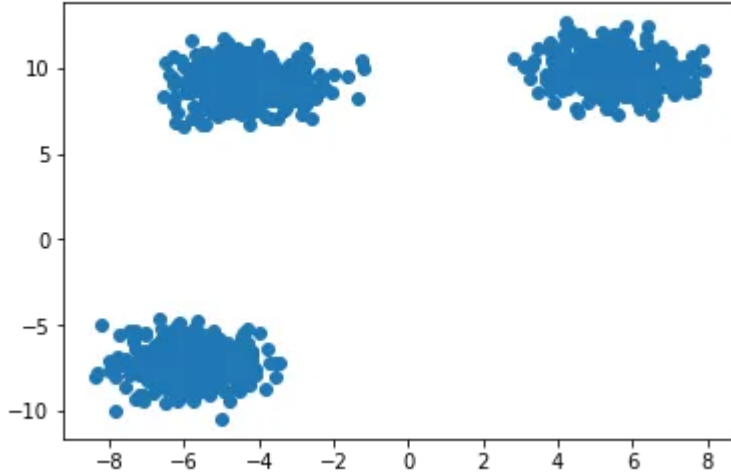
Aufgabe 2

```
körpergrößen = [123, 107, 178, 100, 99, 156, 178, 172, 112, 145, 133, 171, 118]
# Wähle willkürlich zwei Centroide
c1 = körpergrößen[0]
c2 = körpergrößen[1]
cluster1 = []
fertig = False

while not fertig:
    # wiederhole die folgenden Anweisungen
    # bis sich die Cluster nicht mehr ändern
    altesCluster1 = cluster1 # Merke dir c1 aus der vorigen Iteration
    cluster1 = []
    cluster2 = []
    # Die Liste der Körpergrößen wird durchlaufen.
    for g in körpergrößen:
        # Werte, die näher an c1 liegen, werden dem ersten Cluster zugewiesen.
        if (g-c1)**2 < (g-c2)**2:
            cluster1 += [g]
        else:
            # Die anderen Werte werden dem zweiten Cluster zugewiesen.
            cluster2 += [g]
    # Die beiden Centroiden werden neu berechnet
    c1 = sum(cluster1)/len(cluster1) # Mittelwert des ersten Clusters
    c2 = sum(cluster2)/len(cluster2)
    print('Cluster 1:', cluster1)
    print('Cluster 2:', cluster2)
    fertig = cluster1 == altesCluster1 # fertig ist True, wenn sich cluster1 nicht geändert hat.
print('Größe 1:', round(c1), 'cm,',
      'Größe 2:', round(c2), 'cm')
input()
```

Wie bestimmt man das optimale k?

Ellbogenmethode



WSS: Within-Cluster-Sum of Squared Errors

Rückblick

- Menschen und Dinge kann man anhand von Eigenschaften klassifizieren.
- Mit einem Entscheidungsbaum kann man Objekte einer Klasse erkennen.
- Nach dem k-Means-Algorithmus können Objekte aufgrund von messbaren Eigenschaften automatisch in Klassen eingeteilt werden (unüberwachtes Lernen).