h-da HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Einführung in die künstliche Intelligenz

EKI09 – Unsupervised Learning

Prof. Dr. A. del Pino

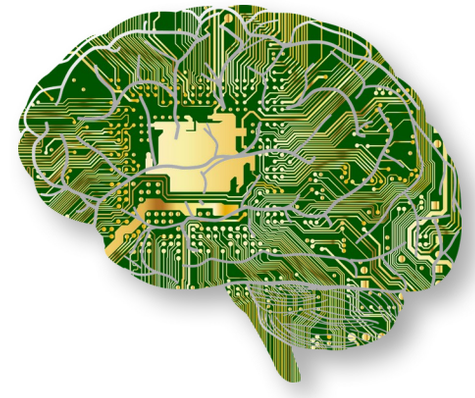


Bild: https://en.wikipedia.org/wiki/Machine_learning#/media/File:Anatomy-1751201_1280.png

Dieser Foliensatz basiert teilweise auf Kursmaterial von Prof. Dr. F. Heinrichs

Aufbau

- Überblick
- Dimensionsreduktion
- Hauptkomponentenanalyse
- Autoencoder
- Clustering
- Clustering Algorithmen
- Mini-Test

Die KI-Landkarte

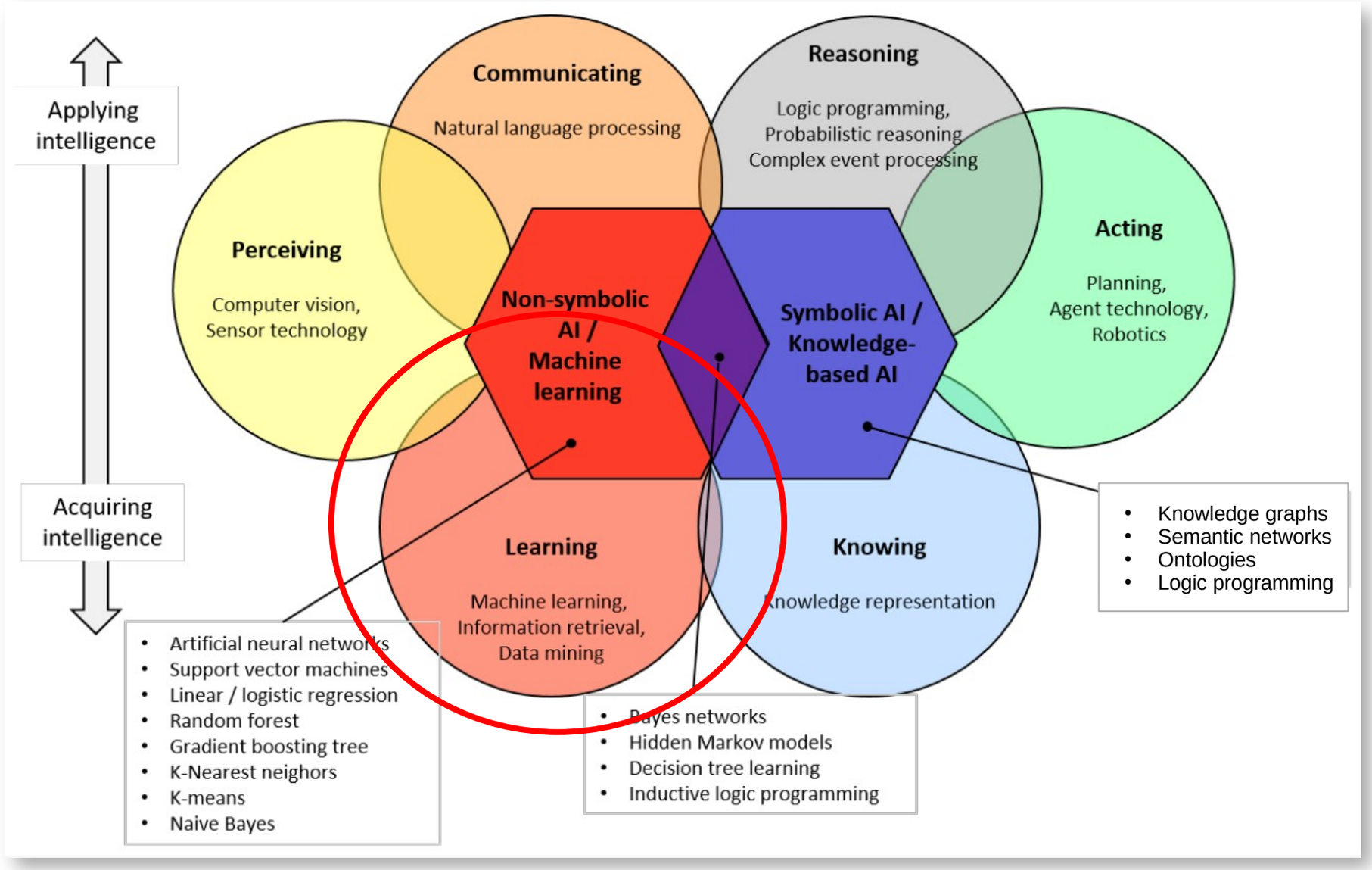


Bild: B. Humm “Applied Artificial Intelligence”, S. 4

Unsupervised Learning

- Unsupervised learning beschäftigt sich mit Datensätzen, über die wenig bekannt ist und die insbesondere kein Label besitzen. Beispiele:

  1.) Gewicht, systolischer und diastolischer Blutdruck und HbA1c-Wert von 492,500 Personen

  2.) Uhrzeit, Anzahl der gekauften Artikel und Gesamtbetrag von 2,740,200 Transaktionen von einem E-Commerce Unternehmen

  3.) Kursverläufe der S&P 500 Konzerne über einen Zeitraum von einem Jahr

  4.) Genexpressionsdaten von 97 Zelltypen mit je 2000 Genen

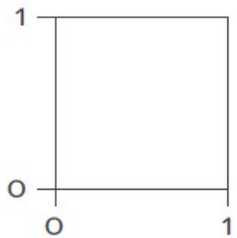
- Was wollen wir über solche Daten wissen?
  - Ausreißererkennung - Gibt es Daten, die stark von den anderen abweichen?
  - Clustering - Kann der gesamte Datensatz sinnvoll in wenige Untergruppen auseinandergebrochen werden?
  - Dimensionsreduktion - Gibt es Daten, die besonders ähnlich zueinander sind?
  - Datengenerierung – Wie sehen ähnliche Daten aus?

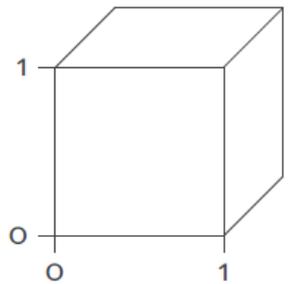
Aufbau

- Überblick
- Dimensionsreduktion
- Hauptkomponentenanalyse
- Autoencoder
- Clustering
- Clustering Algorithmen
- Mini-Test

Dimensionsreduktion

- Warum wollen wir die Dimension von Daten überhaupt reduzieren?
- Was ist das Volumen der folgenden Objekte?
- Was ist der maximale Abstand zwischen zwei Punkten in den Objekten?





$V_1 = 1$ $V_2 = 1$ $V_3 = 1$

$d_1 = 1$ $d_2 = \sqrt{2}$ $d_3 = \sqrt{3}$

allgemein: $V_n = 1$ und $d_n = \sqrt{n}$

Dimensionsreduktion

- Allgemein $V_n = 1$ und $d_n = \sqrt{n}$
- Je höher die Dimension, desto größer die mögliche Distanz zwischen zwei Punkten (in Objekten mit selben Volumen)
- Weiterer Effekt: Wenn wir die Seitenlänge verdoppeln, wächst der Raum exponentiell

$$V([0,1]) = 1 \qquad V([0,2]) = 2$$

$$V([0,1]^2) = 1 \qquad V([0,2]^2) = 4 = 2^2$$

$$V([0,1]^n) = 1 \qquad V([0,2]^n) = 2^n$$

- Statistische Analysen werden schwieriger in hohen Dimensionen

Curse of dimensionality

(Fluch der Dimensionalität)

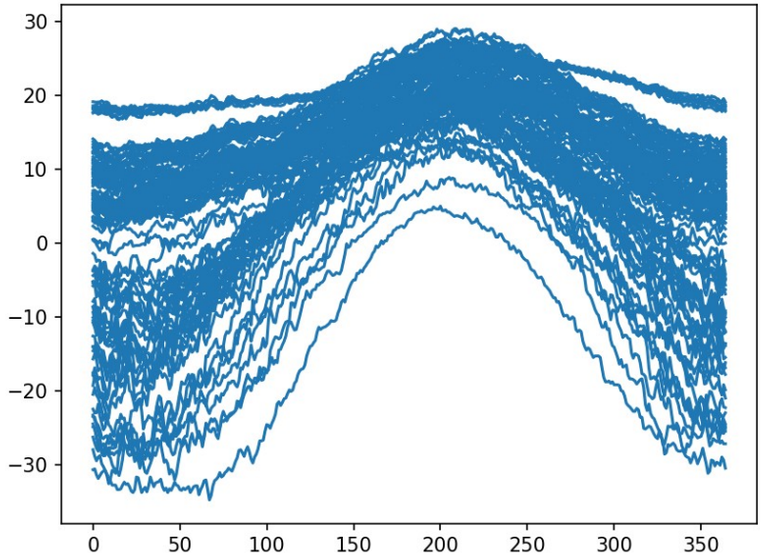
- R. Bellman

Dimensionsreduktion - Beispiel

- Sei $x=(x_1, x_2, \ldots, x_p) \in \mathbb{R}^p$
- Wie können wir die Dimension von x reduzieren?
- Gesucht: Projektion $\phi: \mathbb{R}^p \rightarrow \mathbb{R}^d$ mit $d < p$

Beispiel

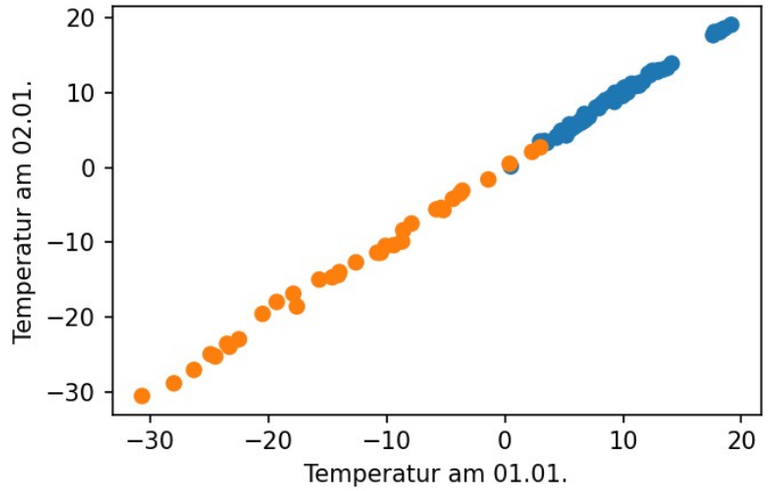
- Temperaturkurven (73 spanische und 35 kanadische Wetterstationen)
- Jede Kurve ist gegeben durch 365 Werte, also 365-dimensional ($\mathbb{R}^{365}$)
- Wie können wir die Dimension reduzieren, z.B. auf $d=2$?

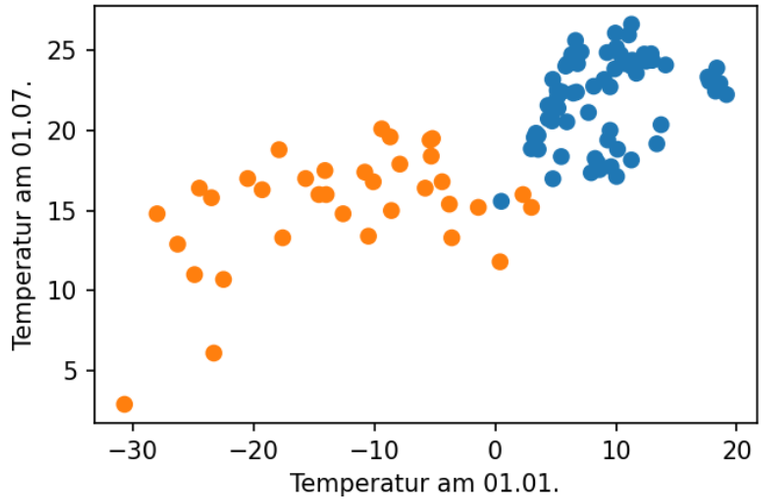


Dimensionsreduktion – Daten weglassen

- Sei $x=(x_1, x_2, \ldots, x_p) \in \mathbb{R}^p$
- Wie können wir die Dimension von x reduzieren?
- Gesucht: Projektion $\phi: \mathbb{R}^p \rightarrow \mathbb{R}^d$ mit $d < p$

- Daten weglassen $\phi(x)=(x_1, x_2, \ldots, x_d)$
- Beispiel: Temperaturwerte von zwei (statt 365 Tagen)
  - Welche Tage?
  - 1. und 2. Januar?
  - 1. Januar und 1. Juli?
- Ggf. hoher Datenverlust
- Bei offensichtlich redundanten Daten sinnvoll, z.B.
  (Temp. in °C, Temp. in °F) → (Temp. in °C)
  (Größe, Gewicht, BMI) → (Größe, Gewicht)

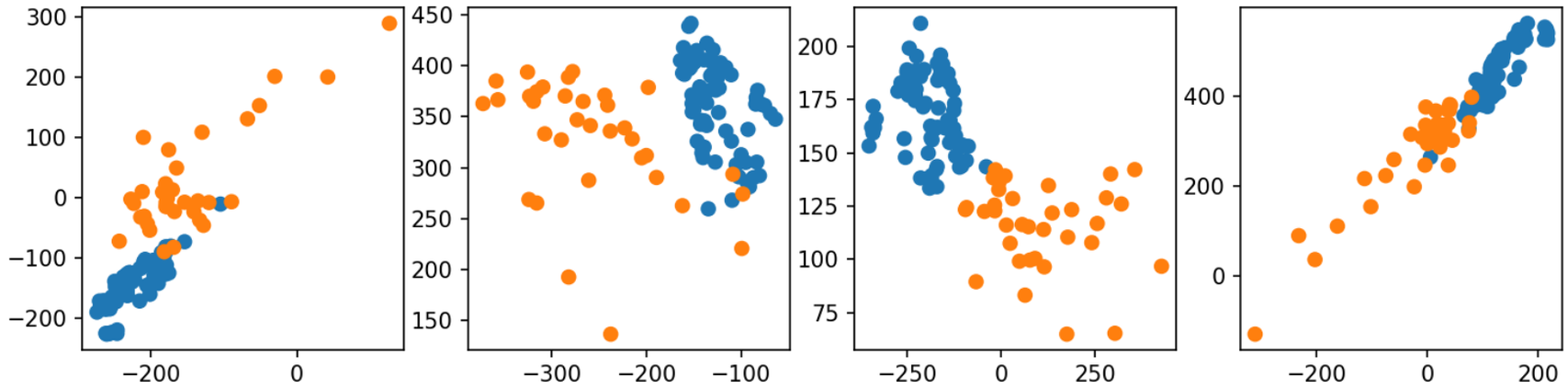




Dimensionsreduktion – Zufallsprojektion

- Sei $x=(x_1, x_2, \ldots, x_p) \in \mathbb{R}^p$
- Wie können wir die Dimension von x reduzieren?
- Gesucht: Projektion $\phi: \mathbb{R}^p \rightarrow \mathbb{R}^d$ mit $d < p$

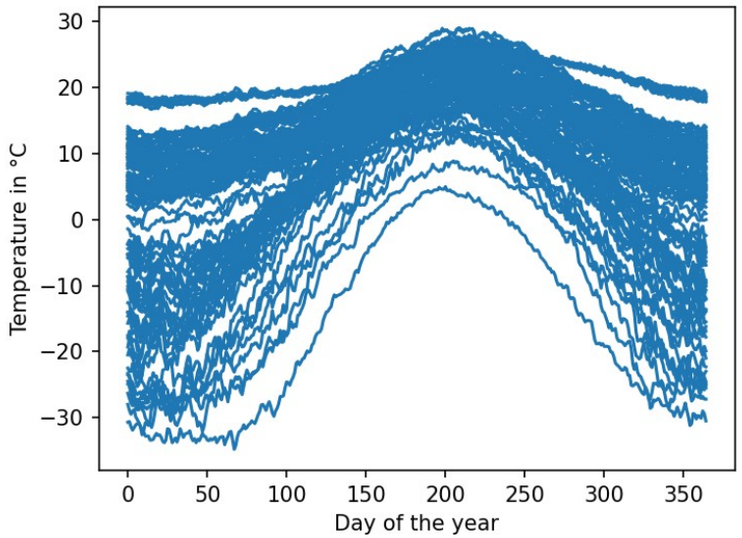
- Für jede Matrix $R \in \mathbb{R}^{p \times d}$, ist $\phi(x) = R\,x$ eine Projektion von $\mathbb{R}^p$ auf $\mathbb{R}^d$
- Zufällige Projektion: Wähle eine zufällige Matrix R und projiziere den Datensatz

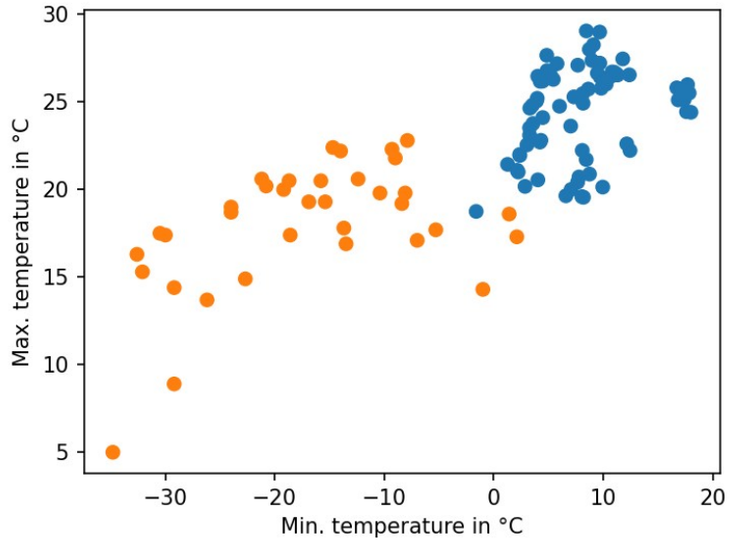


Dimensionsreduktion – Feature Engineering

- Sei $x=(x_1, x_2, \ldots, x_p) \in \mathbb{R}^p$
- Wie können wir die Dimension von x reduzieren?
- Gesucht: Projektion $\phi: \mathbb{R}^p \rightarrow \mathbb{R}^d$ mit $d < p$

- Feature Engineering: Erstelle neue Feature
- $$\phi(x_1, x_2, \ldots, x_p) = \left(\min_{i=1,\ldots,p} x_i, \max_{i=1,\ldots,p} x_i \right)$$
- z.B. minimale und maximale Temperatur im Jahr





Dimensionsreduktion – Fazit

- Variablen weglassen

  - kann zum Verlust wichtiger Informationen führen

- Zufällige Projektionen

  - kann zu guten oder schlechten Projektionen führen, aber nicht vorhersagbar

- Feature Engineering

  - Zur Erstellung sinnvoller Features wird Expertenwissen benötigt

Gibt es Methoden, die selbstständig, datenbasiert Projektionen lernen?

Ja, zum Beispiel:

- Hauptkomponentenanalyse
- Autoencoder

Aufbau

- Überblick
- Dimensionsreduktion
- Hauptkomponentenanalyse
- Autoencoder
- Clustering
- Clustering Algorithmen
- Mini-Test

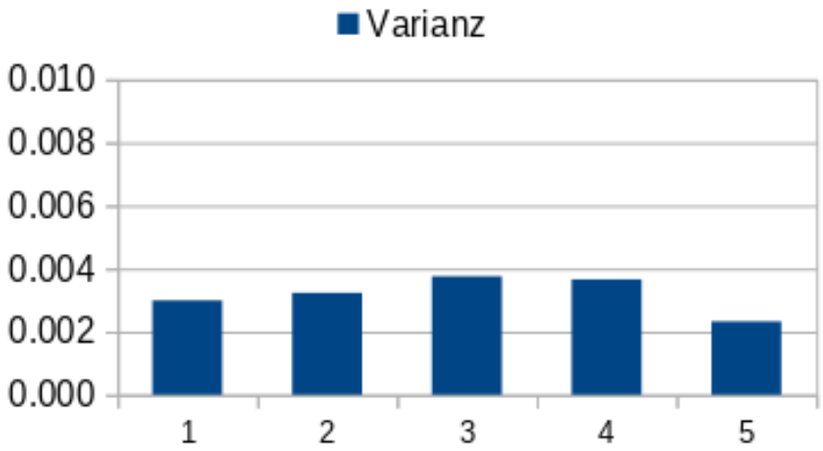
Hauptkomponentenanalyse (principal component analysis, PCA)

ID	f1	f2	f3	f4	f5
1	1.0272	1.0864	0.9767	0.9971	0.9904
2	0.9772	1.1094	0.8886	1.1491	1.0625
3	1.0270	1.0944	0.9652	1.0935	1.0689
4	1.1549	0.9138	1.0394	1.0025	1.0365
5	1.0197	1.0236	0.9551	1.1626	1.0450
6	1.0541	1.0882	0.8779	1.0969	0.9224
7	0.9586	1.0854	0.8438	1.1411	0.9913
8	1.0371	1.0550	0.8532	1.0274	1.0684
9	0.9955	1.0842	0.9164	1.1263	1.0594
10	1.0689	1.0609	0.9372	1.0884	0.9909
Varianz	0.00299	0.00323	0.00376	0.00366	0.00233

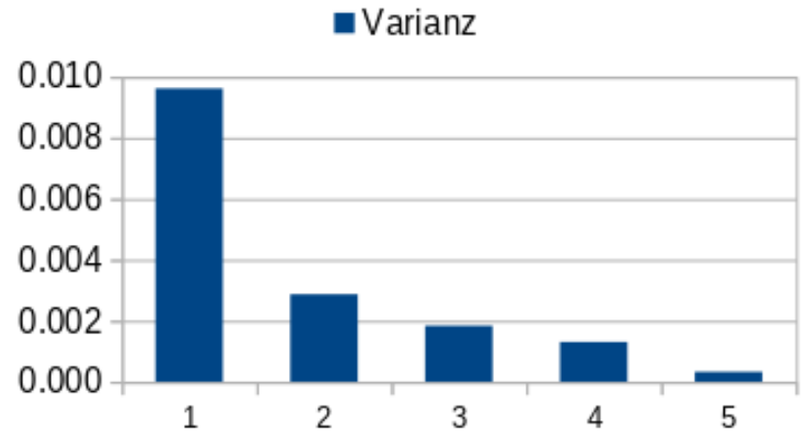
PCA

ID	PC1	PC2	PC3	PC4	PC5
1	0.05273	0.05702	0.04892	-0.06151	0.02424
2	-0.09926	-0.04378	0.01153	-0.00328	-0.00800
3	-0.00019	-0.04359	0.02989	-0.04109	-0.01976
4	0.23564	-0.02074	-0.01981	0.02584	0.00665
5	-0.00639	-0.07264	-0.05370	0.00019	0.01103
6	-0.03297	0.09826	-0.05303	0.00132	-0.01674
7	-0.11804	0.02119	-0.02223	0.02536	0.03115
8	-0.00362	0.02056	0.07545	0.06939	-0.00711
9	-0.05228	-0.04272	0.01029	-0.00661	-0.00063
10	0.02439	0.02642	-0.02733	-0.00962	-0.02082
Varianz	0.00962	0.00287	0.00184	0.00131	0.00033

$$Varianz(x) = \frac{1}{n-1}\sum_i (x_i - \bar{x})$$



Erklärte Varianz (explained variance)

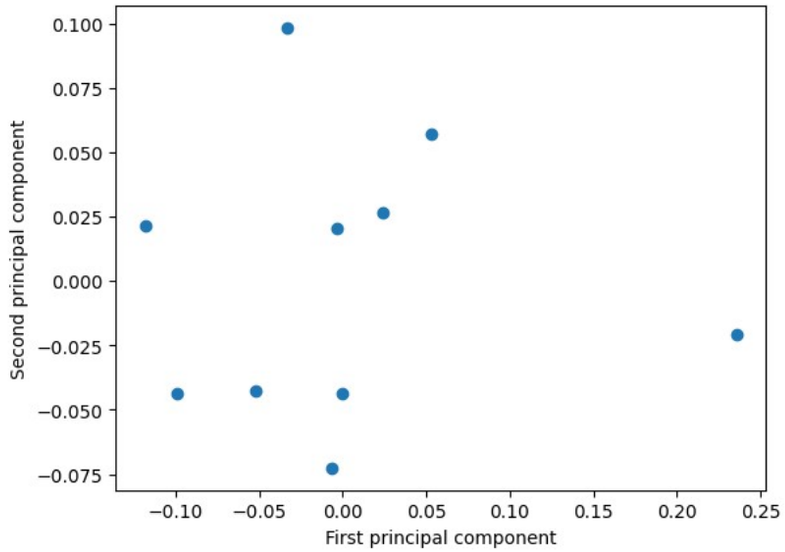


Hauptkomponentenanalyse - Anwendung

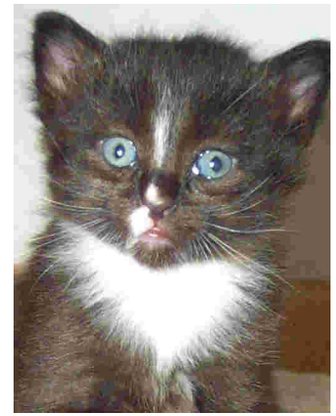
Darf man das überhaupt?

Wir werfen alle Komponenten außer PC1 und PC2 weg, da sich die meiste Varianz der Daten nun in PC1 und PC2 befindet. Damit können wir einen 2-dimensionalen Plot erstellen, der vieles von den Daten erklärt.

ID	PC1	PC2	PC3	PC4	PC5
1	0.05273	0.05702	0.04892	-0.06151	0.02424
2	-0.09926	-0.04378	0.01153	-0.00328	-0.00800
3	-0.00019	-0.04359	0.02989	-0.04109	-0.01976
4	0.23564	-0.02074	-0.01981	0.02534	0.00665
5	-0.00639	-0.07264	-0.05370	0.00019	0.01103
6	-0.03297	0.09826	-0.05303	0.00182	-0.01674
7	-0.11804	0.02119	-0.02223	0.02536	0.03115
8	-0.00362	0.02056	0.07045	0.06939	-0.00711
9	-0.05228	-0.04272	0.01029	-0.00661	-0.00063
10	0.02439	0.02642	0.02733	-0.00962	-0.02083
Varianz	0.00962	0.00287	0.00184	0.00131	0.00033



Warum nicht?



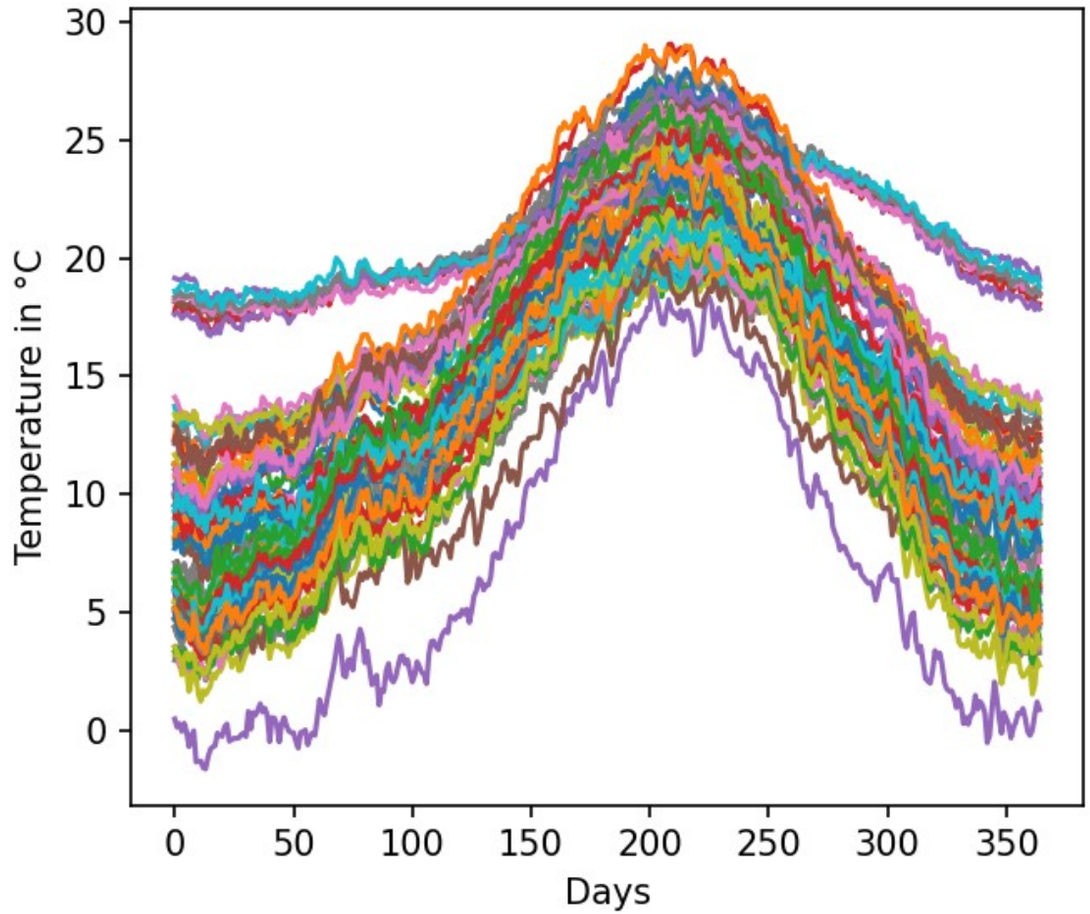
Hauptkomponentenanalyse – Algorithmus

- Normalisiere die Features, so dass der Erwartungswert für jedes Feature null ist
- Berechne aus den n normalisierten Features die Kovarianzmatrix
  - Symmetrische $n \times n$ Matrix
- Berechne für diese Matrix die n Eigenvektoren und ihre zugehörigen Eigenwerte $M x = \lambda x, \quad x \neq 0$
- Sortiere die Eigenvektoren nach absteigender Größe ihrer Eigenwerte
- Bilde nun Linearkombinationen aus den Bestandteilen der Eigenvektoren und der Features
  - Die Werte aus dem Eigenvektor mit dem größten Eigenwert bilden die Koeffizienten a_i mit denen die erste Hauptkomponente gebildet wird $PC1 = a_1 * f1 + a_2 * f2 + ... + a_n * fn$
  - Die Werte aus dem Eigenvektor mit dem zweitgrößten Eigenwert bilden die Koeffizienten b_i mit denen die zweite Hauptkomponente gebildet wird $PC2 = b_1 * f1 + b_2 * f2 + ... + b_n * fn$
  - …
  - Die Werte aus dem Eigenvektor mit dem kleinsten Eigenwert bilden die Koeffizienten mit denen die letzte Hauptkomponente gebildet wird

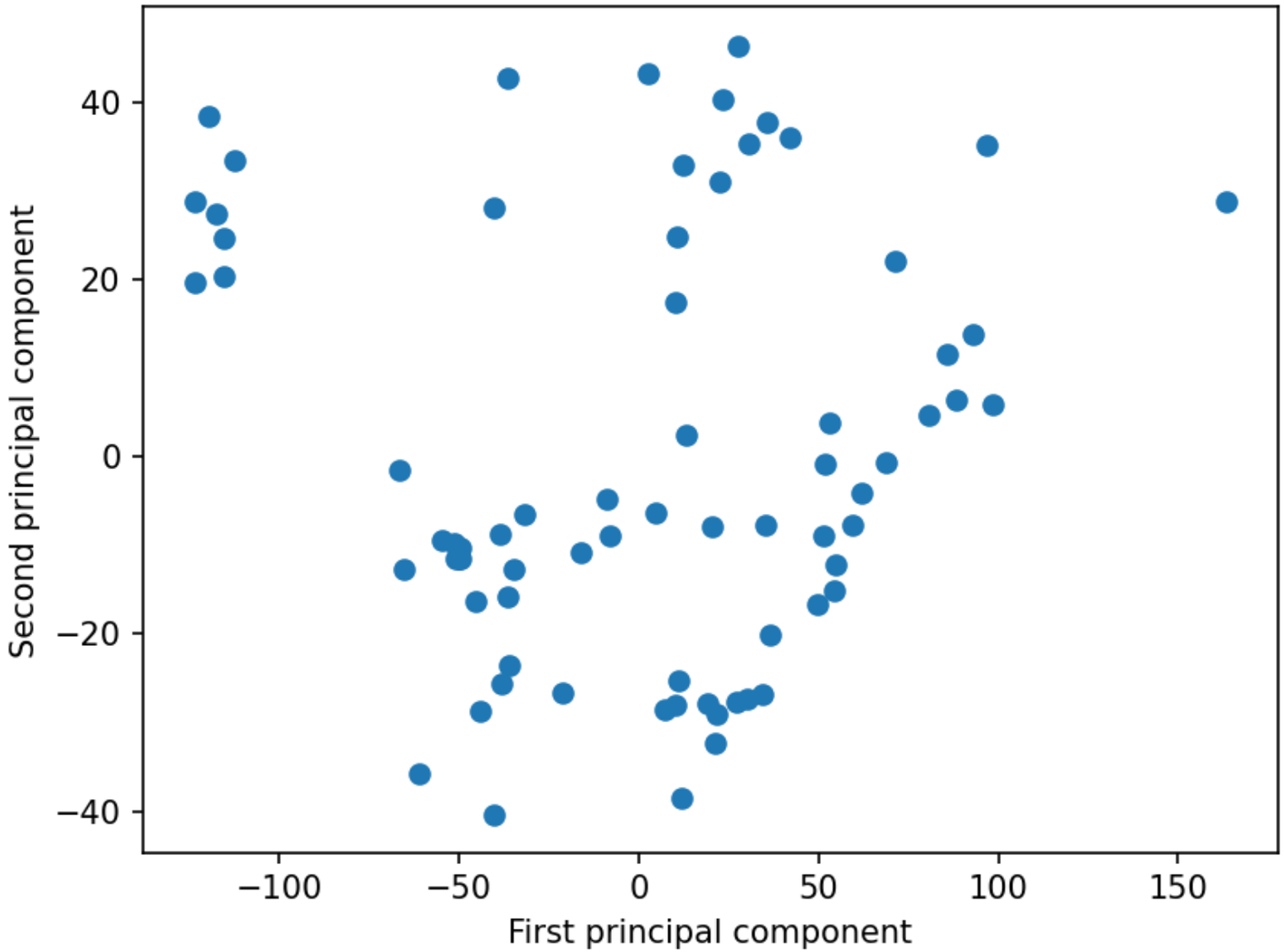
Hauptkomponentenanalyse – Beispiel

- Temperaturkurven (73 spanische Wetterstationen)
- Jede Temperaturkurve ist 365-dimensional
- Projektion auf zwei Komponenten

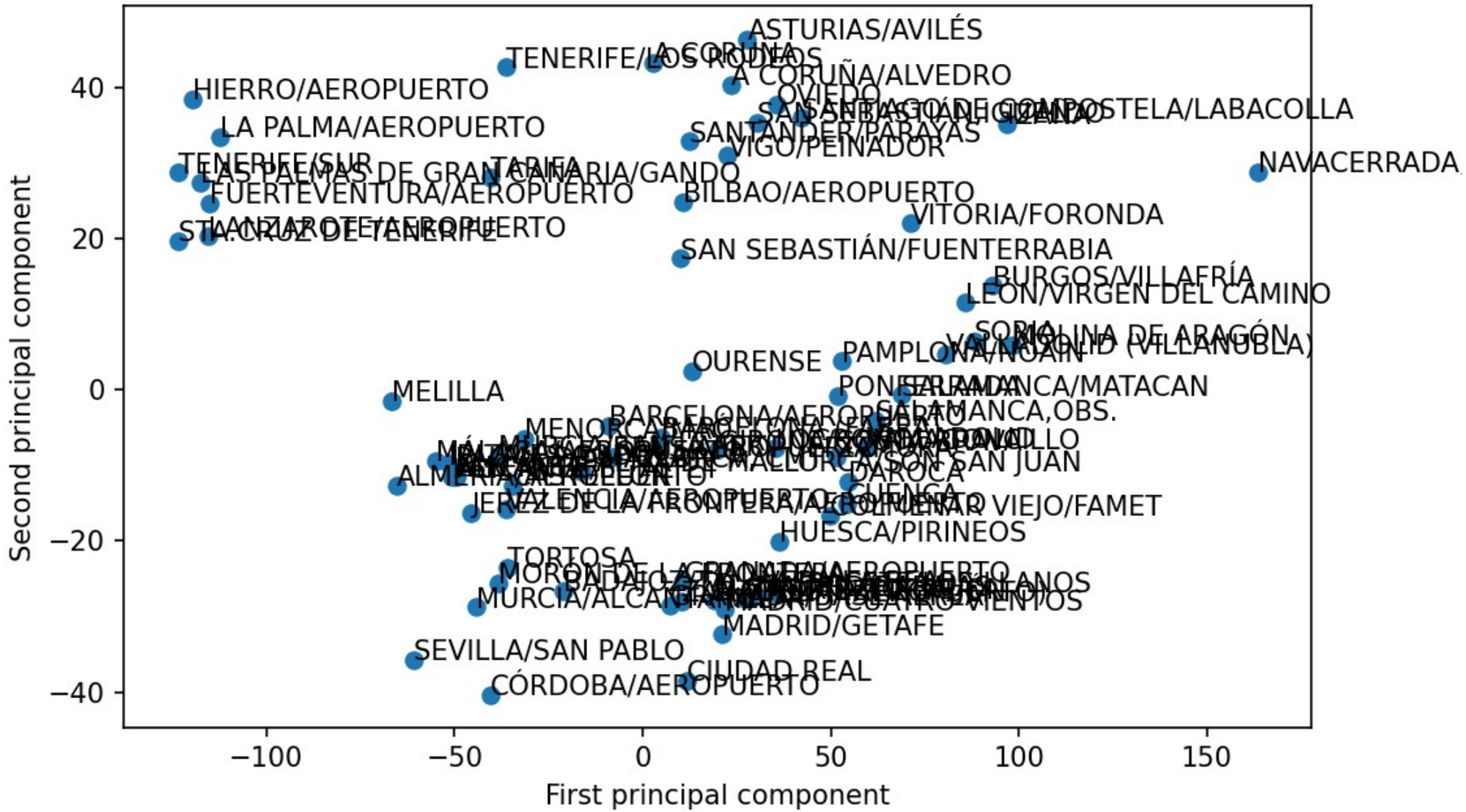
```
from sklearn.decomposition import PCA

model = PCA(n_components=2)

model.fit(X)

X_proj = model.transform(X)
```



Hauptkomponentenanalyse – Beispiel

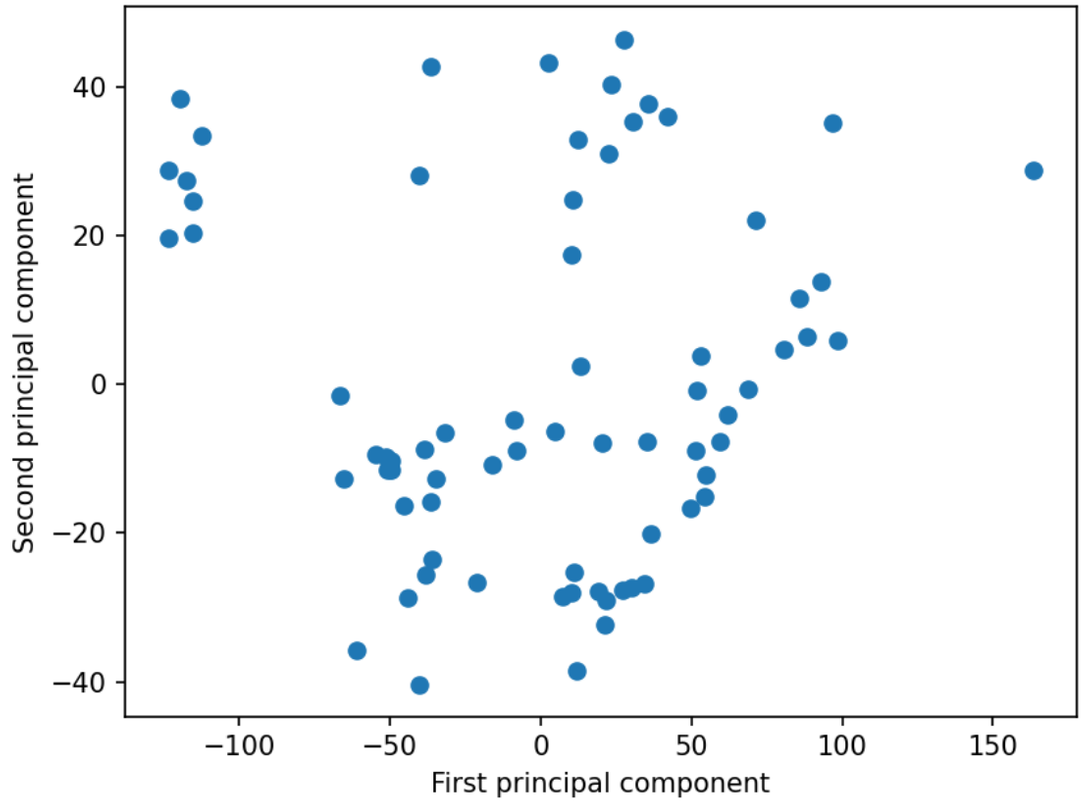


Hauptkomponentenanalyse – Beispiel



Hauptkomponentenanalyse – Beispiel

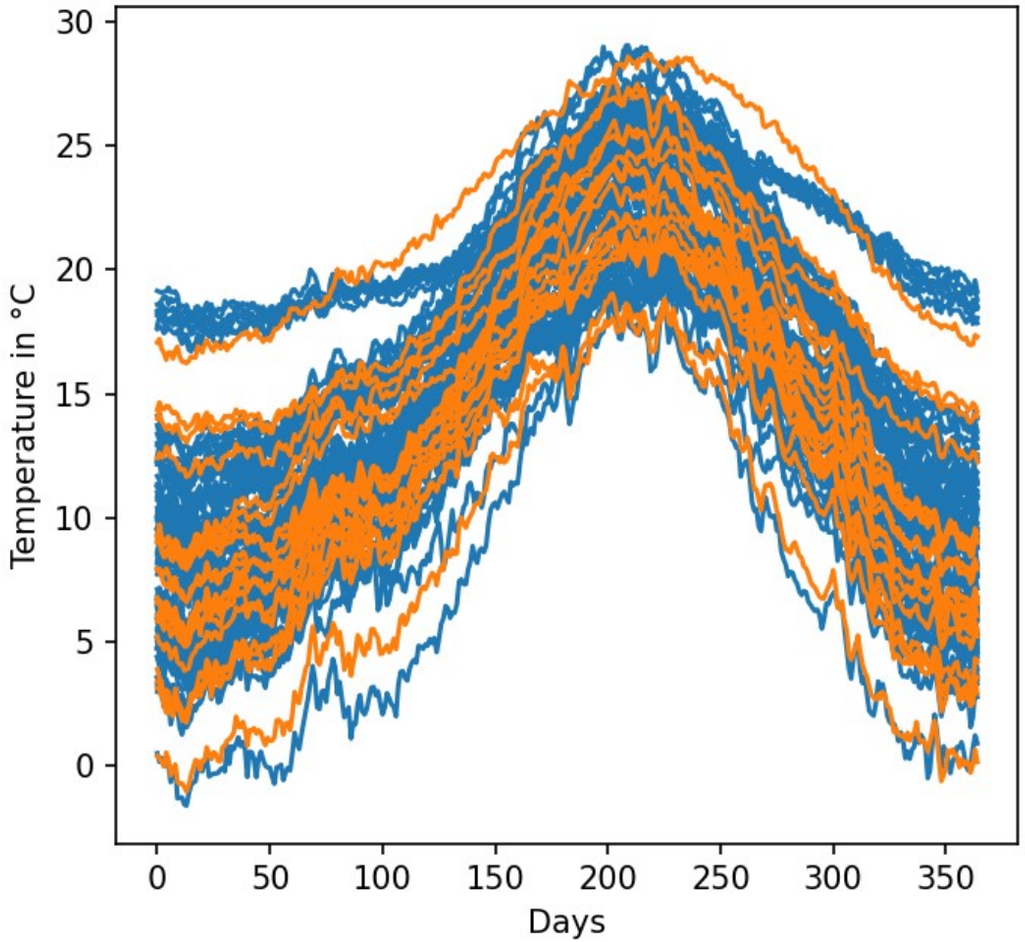
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA

def load_data() -> np.ndarray:
    filepath = "temperature_data_es.csv"
    df = pd.read_csv(filepath).to_numpy().transpose()
    return df

if __name__ == '__main__':
    X = load_data()
    model = PCA(n_components=2)
    X_proj = model.fit_transform(X)
    plt.scatter(X_proj[:, 0], X_proj[:, 1])
    plt.xlabel("First principal component")
    plt.ylabel("Second principal component")
    plt.show()
```

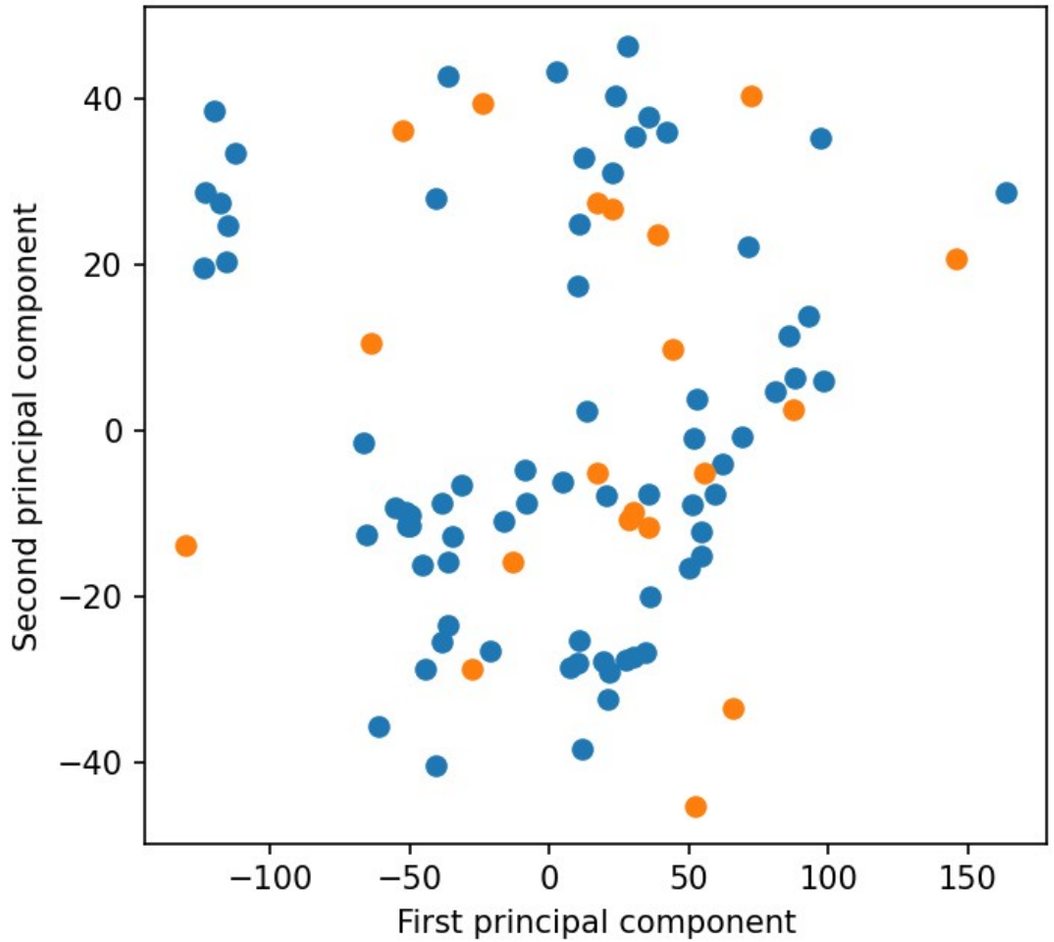



```
model.fit(x)
X_proj = model.transform(X)
```

Hauptkomponentenanalyse – Eine weitere Anwendung

Wir können mit der Hauptkomponentenanalyse neue Daten erzeugen:





Aufbau

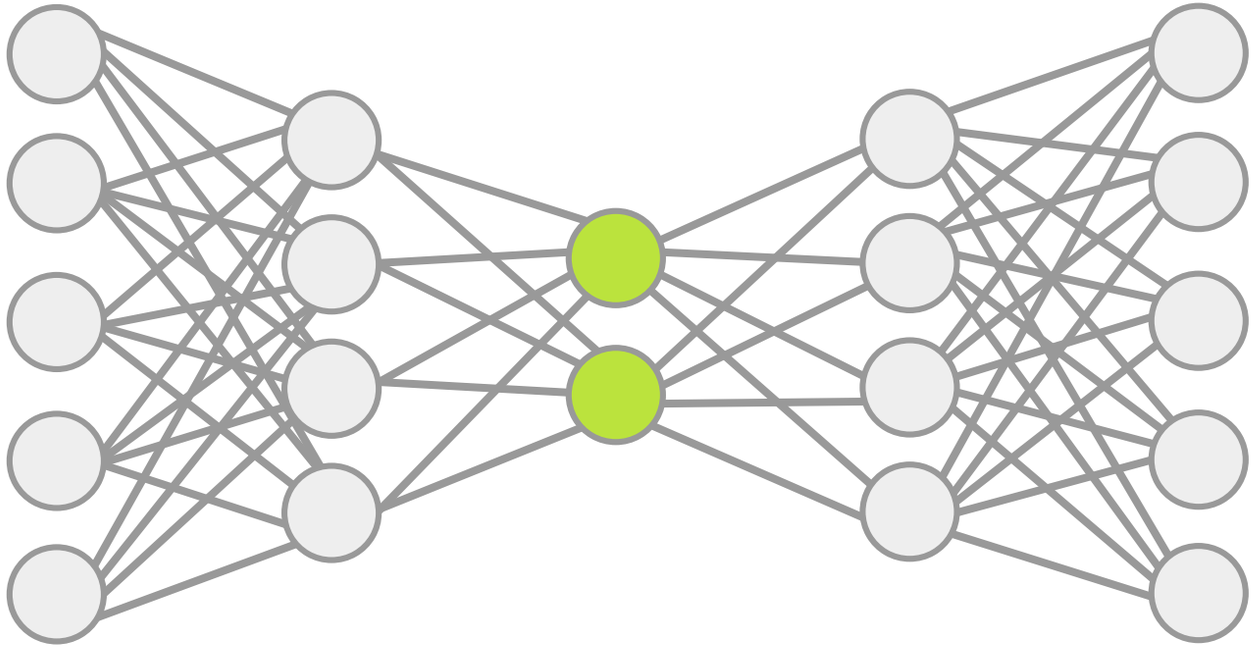
- Überblick
- Dimensionsreduktion
- Dimensionsreduktion mit PCA
- Autoencoder
- Clustering
- Clustering Algorithmen
- Mini-Test

Autoencoder

Originale Daten

x

Input



(Möglichst gute) Approximation der originalen Daten

$\hat{x}$

Output

Encoder

Latent Space

Decoder

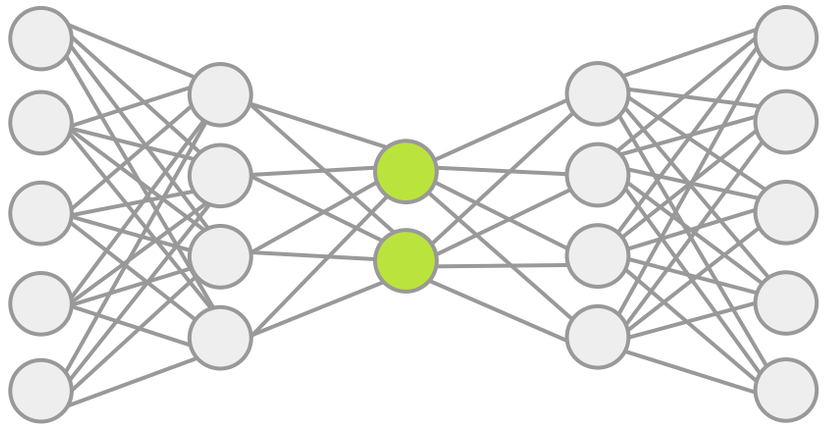
Projektion von Input in Latent Space (Dimensionsreduktion)

Niedrig-dimensionaler Raum

Transformation vom Latent Space zu Output. (Unvollständige Rekonstruktion)

Autoencoder – Anwendungen

- Dimensionsreduktion
  - Nicht-linear ggü. linearer PCA
- Entfernung von Rauschen *(noise reduction)*
  - Training mit Daten, denen Rauschen hinzugefügt wurde. Die Kostenfunktion minimiert die Differenz zwischen dem Output und dem Originalbild ohne Rauschen.
- Ausreißererkennung *(outlier detection)*
- Datengenerierung *(data generation)*
  - Zufälliger Vektor im Latent Space

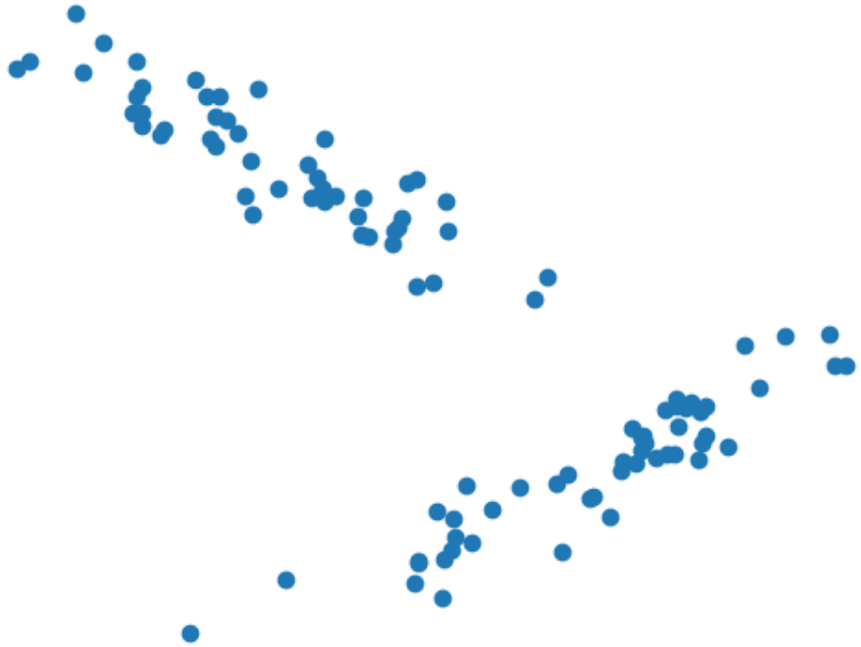


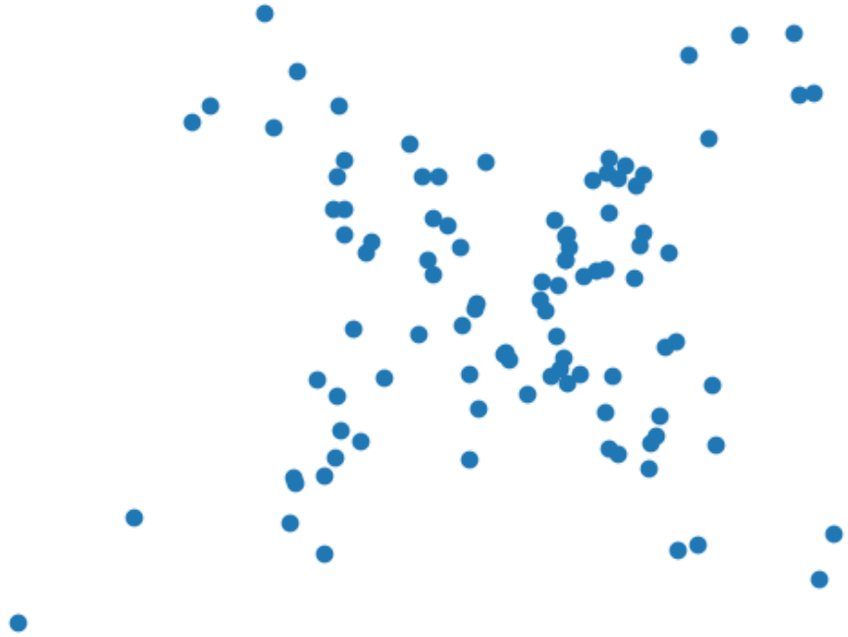
Aufbau

- Überblick
- Dimensionsreduktion
- Dimensionsreduktion mit PCA
- Autoencoder
- Clustering
- Clustering Algorithmen
- Mini-Test

Clustering

Gibt es Gruppen von Punkten, sodass Beobachtungen innerhalb von Gruppen ähnlicher/näher sind als Punkte verschiedener Gruppen?





Clustering – Beispiel Kundensegmentierung (Customer Segmentation)

Warum Clustering?

- Um Aussagen über Gruppen in Datensatz zu machen (z.B. Kundensegmente)

Ziel:

- Personalisierung von Werbung
- Erhöhung von Kundenzufriedenheit und -loyalität

Daten:

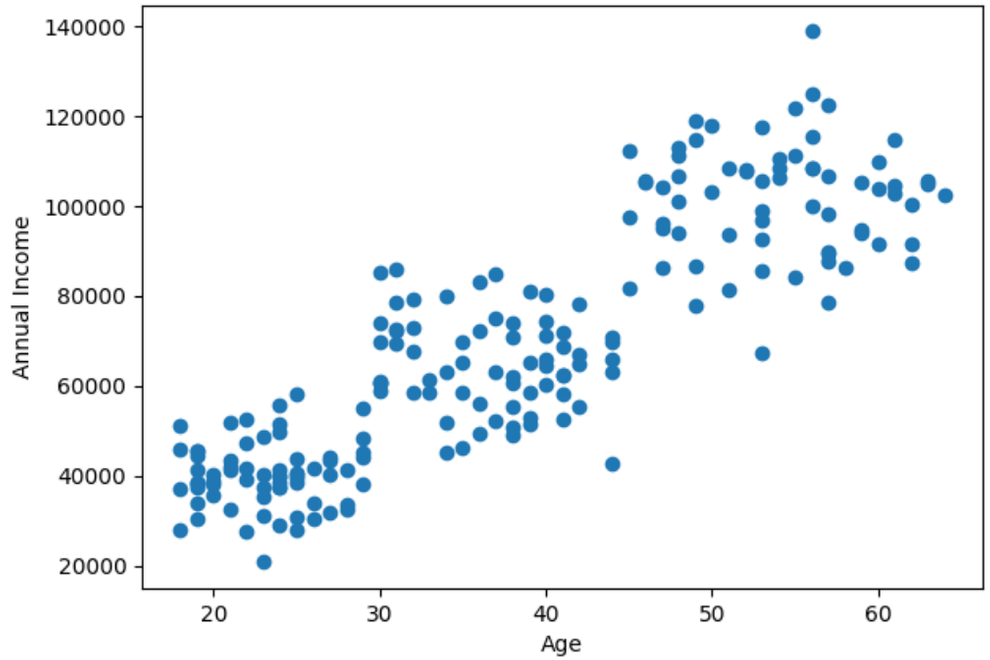
- Alter und jährliches Einkommen von 201 Kunden

Ergebnis:

- Zuordnung der Kunden zu Kundensegmenten

Weitere mögliche Daten:

- Online-Verhalten
- Demografische Daten
- ...

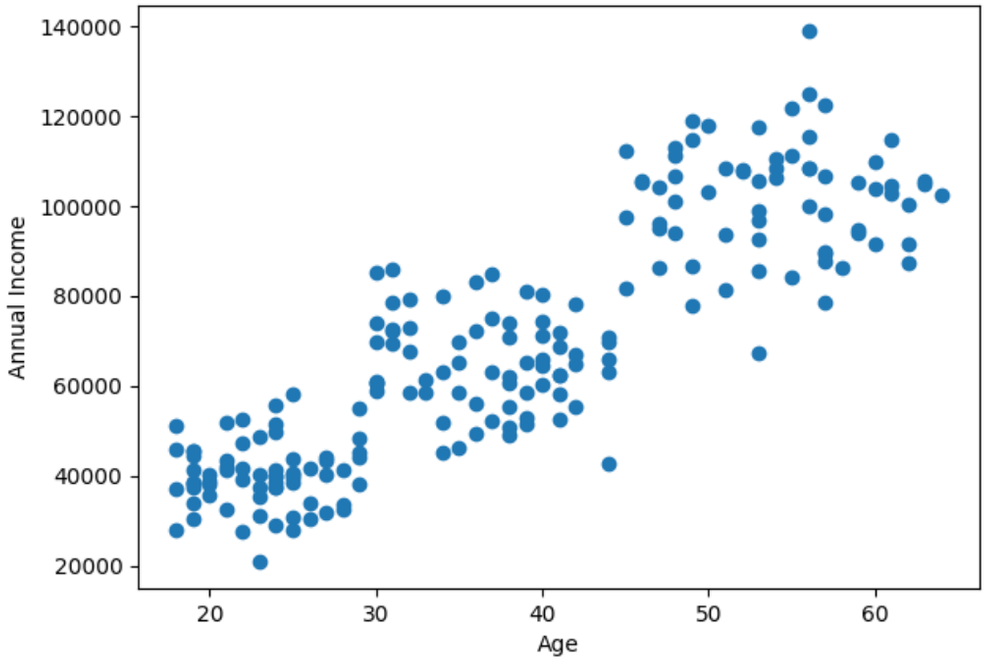


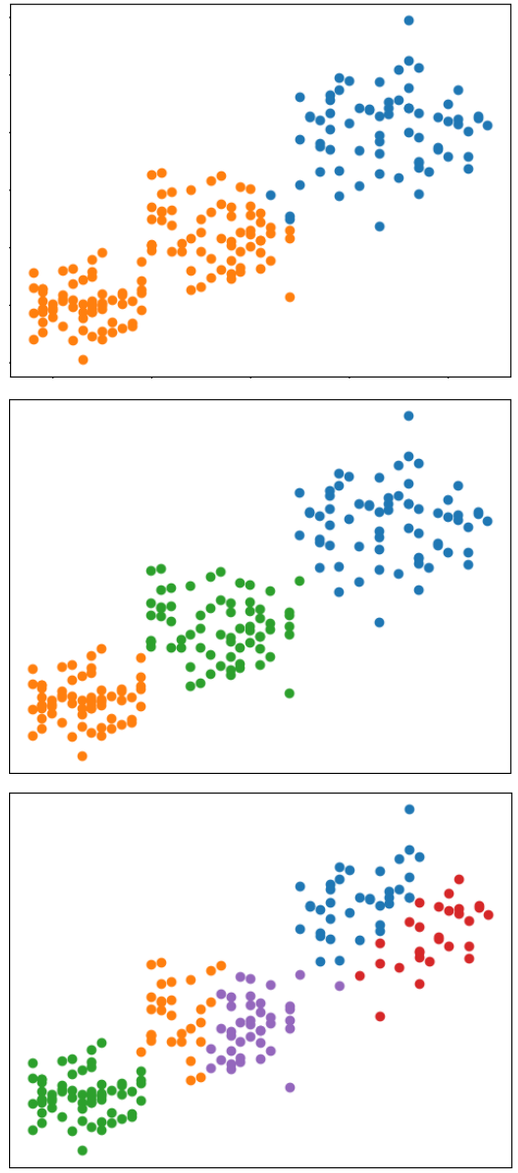
Kundensegmente sind nicht a priori gegeben!

Clustering

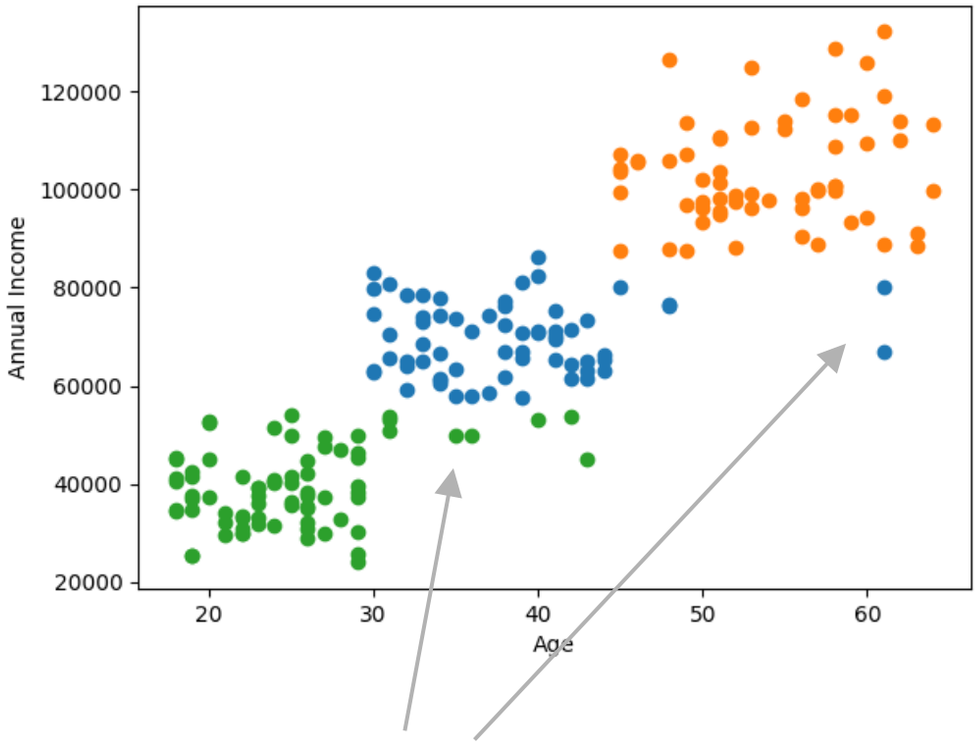
Wie viele Cluster gibt es?

Welche Anwendungen hat die Clusteranalyse?





Clustering



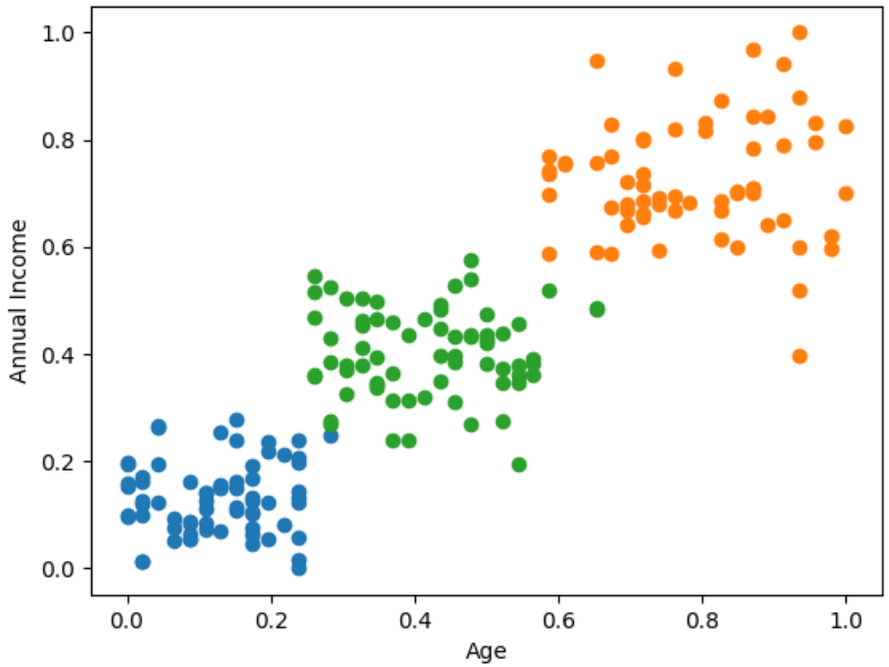
Sind die Cluster gut?

- K-Means nutzt euklidischen Abstand.

  → Alter 30 und 60 sind näher als Einkommen 50000 und 50031

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans

filepath = "customer_segmentation.csv"
df = pd.read_csv(filepath)

num_clusters = 3
model = KMeans(n_clusters=num_clusters)
model.fit(df)
df['Cluster'] = model.predict(df)

for cluster in range(num_clusters):
    cluster_data = df[df['Cluster'] == cluster]
    plt.scatter(cluster_data['Age‘],
                        cluster_data['Annual_Income'])

plt.xlabel('Age')
plt.ylabel('Annual Income')
plt.show()
```

Clustering



Besser!

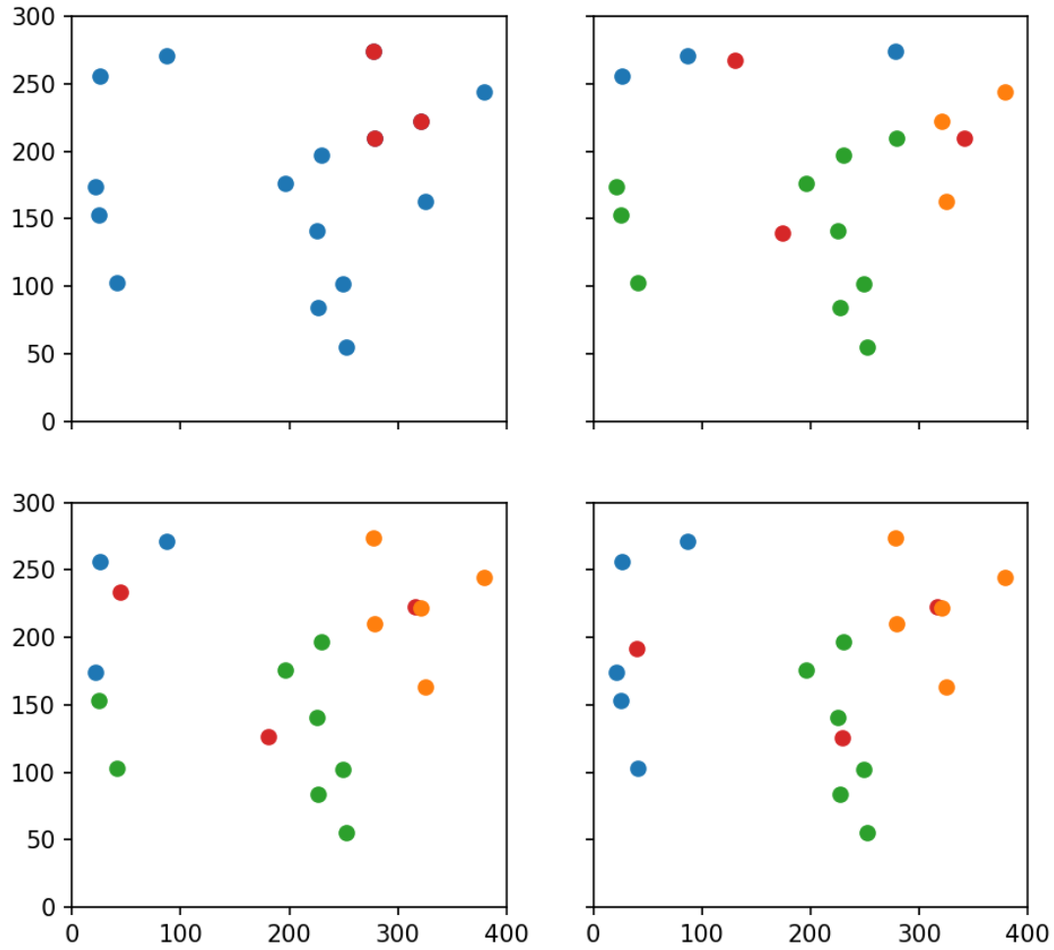
```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans

filepath = "customer_segmentation.csv"
df = pd.read_csv(filepath)

df_norm = (df - df.min()) / (df.max() - df.min())

num_clusters = 3
model = KMeans(n_clusters=num_clusters)
model.fit(df_norm)
df_norm['Cluster'] = model.predict(df_norm)

for cls in range(num_clusters):
    cluster_data = df_norm[df_norm['Cluster'] == cls]
    plt.scatter(cluster_data['Age'],
                        cluster_data['Annual_Income'])

plt.xlabel('Age')
plt.ylabel('Annual Income')
plt.show()
```

Aufbau

- Überblick
- Dimensionsreduktion
- Dimensionsreduktion mit PCA
- Autoencoder
- Clustering
- Clustering Algorithmen
- Mini-Test

K-Means Algorithmus

- Initiale Wahl von zufälligen Clusterzentren
- Bestimmen und iteratives Update der Clusterzentren



K-Means Algorithmus

Gegeben:

- Datensatz $X_1, \ldots X_n$
- Gewünschte Anzahl k der Cluster

Gesucht:

- Cluster $C_1, \ldots, C_k$, mit ihren zugeordneten X_j
- Clusterzentren $m_1, \ldots m_k$ für jedes der k Cluster

K-Means Algorithmus:

1. Wähle zufällig k Datensätze, um die initialen Clusterzentren $m_1, \ldots m_k$ festlegen
2. Ordne jeden Datensatz dem Cluster zu, an dem er am nähesten dran ist

$$C_i = \{ X_j : \|X_j - m_i\|^2 = \min_{p=1,\ldots,k} \|X_j - m_p\|^2 \}$$

3. Berechne für jedes Cluster sein Clusterzentrum aus den zugeordneten Daten

$$m_i = \frac{1}{|C_i|} \sum_{X_j \in C_i} X_j$$

4. Wiederhole Schritte 2+3 bis sich die Clusterzentren m_i nicht mehr verändern

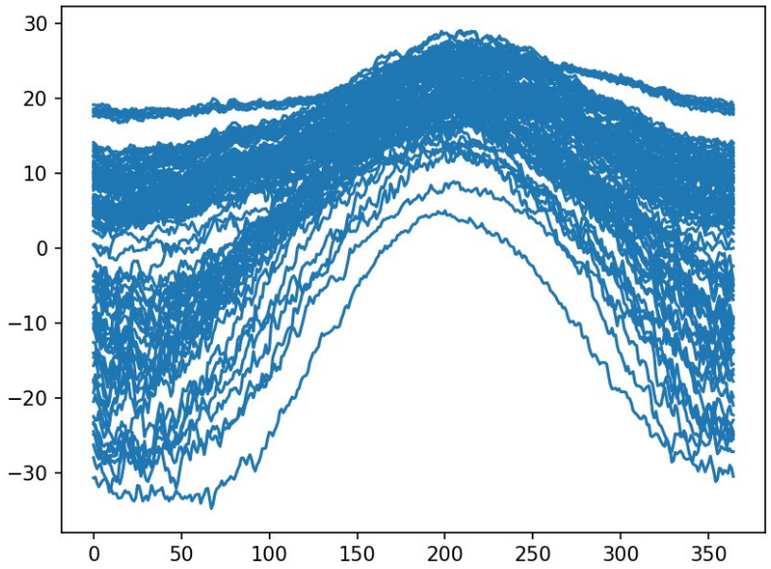
K-Means Algorithmus

Wie finden wir die Cluster?

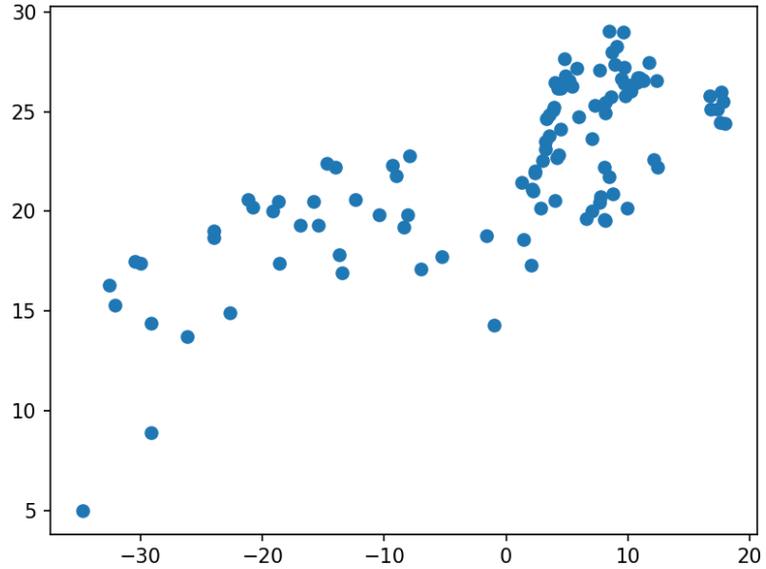
```
from sklearn.cluster import KMeans


model = KMeans(n_clusters=k)

model.fit(data)

prediction = model.predict(data)
```

K-Means Algorithmus – Beispiel

Temperaturkurven von 73 spanischen und 35 kanadischen Wetterstationen



Min./Max.
Temperatur

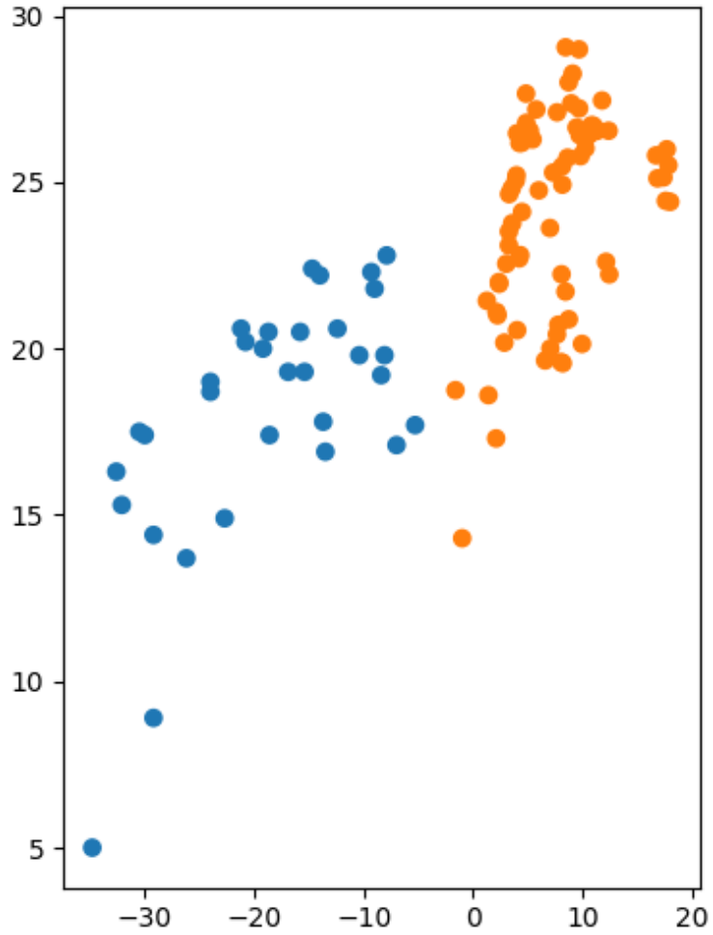



```
from pca_example import load_data

data, labels = load_data('both')
data_2d = np.stack([np.min(data, axis=-1), np.max(data, axis=-1)], axis=-1)
```

K-Means Algorithmus – Beispiel

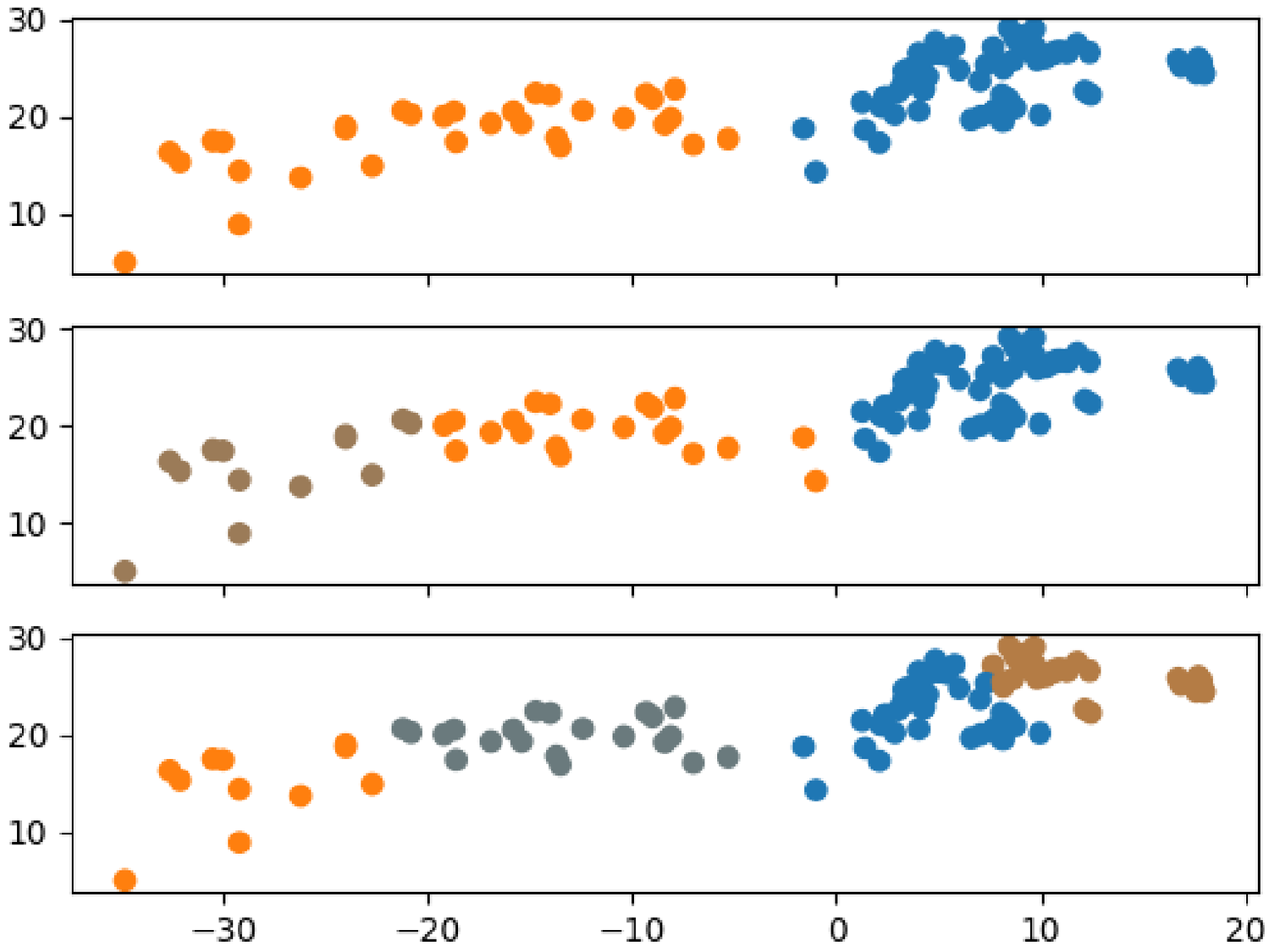
Wie finden wir die Cluster?

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
from pca_example import load_data

if __name__ == '__main__':
    data, labels = load_data('both')
    data_2d = np.stack([np.min(data, axis=-1), np.max(data, axis=-1)], axis=-1)

    fig, axes = plt.subplots(3, sharex=True, sharey=True)
    for k, ax in zip([2, 3, 4], axes):
        model = KMeans(n_clusters=k)
        model.fit(data_2d)
        pred = model.predict(data_2d)
        ax.scatter(data_2d[:, 0], data_2d[:, 1], c=pred)

    plt.show()
```



K-Means Algorithmus – Beispiel

Wie viele Cluster?



K-Means Varianten

- *k*-means wählt die initialen Clusterzentren zufällig (gleichverteilt)

  Ist das sinnvoll? Alternative?

- *k*-means++ wählt Clusterzentren sukzessive mit unterschiedlicher Verteilung
- *k*-medians wählt statt Mittelwert, jeweils den Median pro Cluster
- *k*-medoids erlaubt als Clusterzentren nur Punkte im Datensatz

K-Means++ Algorithmus

- Der *k*-Means++ Algorithmus verwendet eine verbesserte Auswahl der initialen Clusterzentren
- Alles weitere, d.h. das interative Zuordnen der Punkte und die Neuberechnung der Clusterzentren verläuft wie beim K-Means Algorithmus.

K-Means++ Initialisierung:

1. Wähle (gleichverteilt) ein zufälliges Clusterzentrum m_1 aus dem Datensatz

Für $j=2,\ldots,k$

2. Bestimme für jeden Punkt X im Datensatz den Abstand zum nähesten bekannten Clusterzentrum

$$D(X)=\min_{i=1,\ldots,j-1} \|X-m_i\|$$

3. Wähle das nächste Clusterzentrum m_j aus der Menge aller Punkte mit einer Wahrscheinlichkeit proportional zu $D^2(X)$

$$P(X_j)=\frac{D^2(X)}{\sum_{i=1}^{n} D^2(X_i)}$$

Übersicht Clustering-Algorithmen

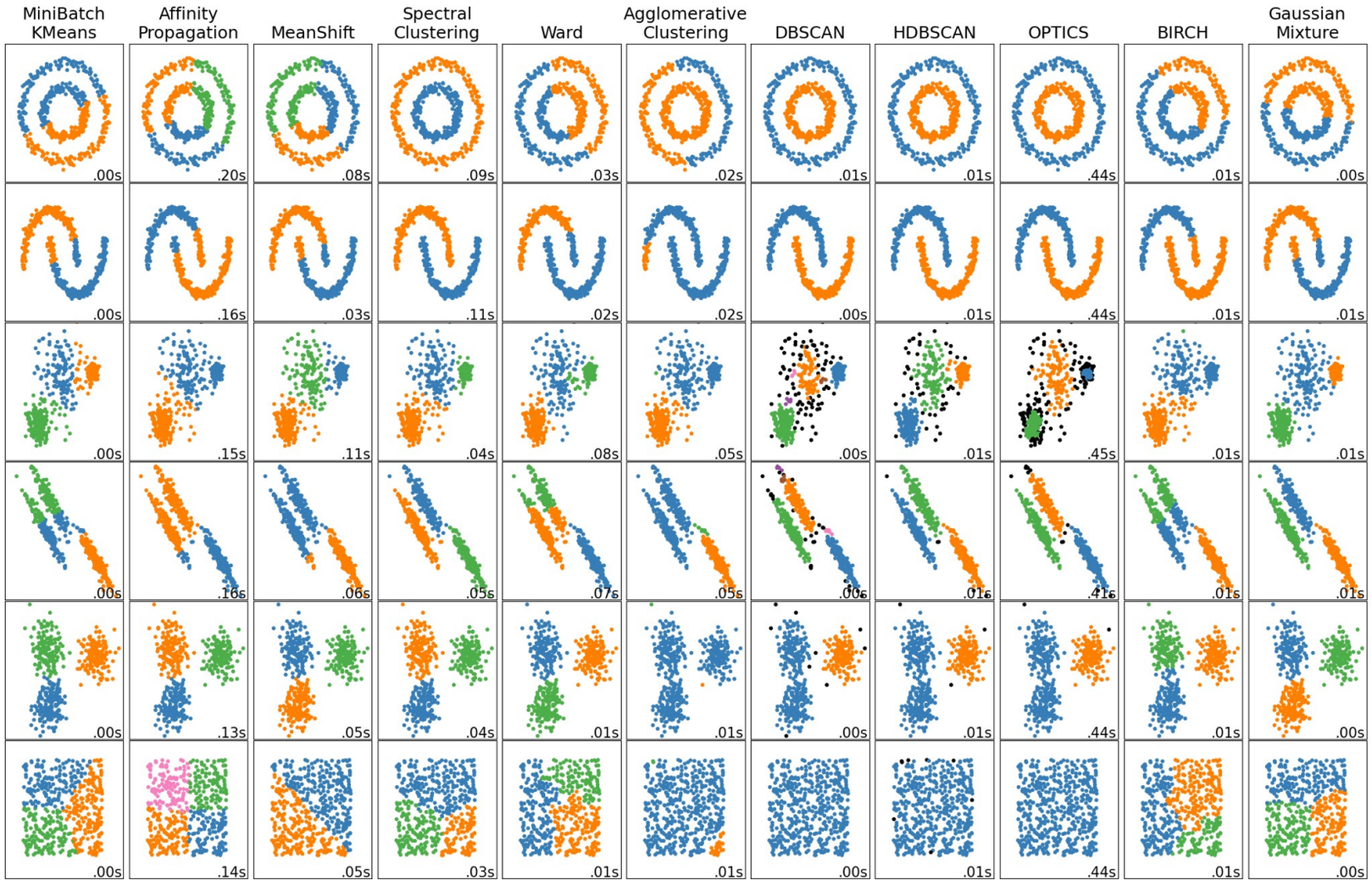


Bild: https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

Mini-Test “Unsupervised Learning”, fällig am (Siehe Moodle)

- Was ist Unsupervised Learning?
- Was ist das Ziel der Clusteranalyse?
- Was sind Ausreißer? Warum wollen wir sie detektieren?
- Wo spielt die Generierung neuer Daten eine Rolle?
- Warum wollen wir die Dimension von Datensätzen reduzieren?
- Welche Möglichkeiten der Dimensionsreduktion gibt es?
- Wie können wir mit Hilfe der Hauptkomponentenanalyse neue Daten erzeugen?
- Wie sind Autoencoder aufgebaut?
- Wie können wir mit Autoencodern die Dimension reduzieren?
- Wie funktioniert der k-Means Algorithmus?