

- **Thema 1 - Polynomial Features (Polynomiale Merkmale) // Mohamed Houria**
- **Thema 3 - Feature Encoding für hohe Kategoriezahlen // Firas Arfaoui**
- **Thema 4 - Feature Engineering für Zeitreihen // Wafaa Arzane**
- **Thema 5 - Vergleich von Skalierungstechniken // Bilal El Arbati**

- **Thema 1 - Polynomial Features (Polynomiale Merkmale) // Mohamed Houria**
- **Thema 3 - Feature Encoding für hohe Kategoriezahlen // Firas Arfaoui**
- **Thema 4 - Feature Engineering für Zeitreihen // Wafaa Arzane**
- **Thema 5 - Vergleich von Skalierungstechniken // Bilal El Arbati**

# Polynomial Features:

- Das sind neue Merkmale, die entstehen, indem man vorhandene Merkmale potenziert (hoch 2, hoch 3 usw.) oder miteinander multipliziert. Dadurch kann das Modell besser erkennen, wie Merkmale zusammenwirken oder nichtlineare Zusammenhänge bestehen. Beispiel: In einem Modell zur Vorhersage von Immobilienpreisen kann Fläche<sup>2</sup> dem Modell mehr Informationen geben, wie größere Häuser den Preis beeinflussen.

# Polynomial Features:

## Vorteile

- Erkennen komplexer Zusammenhänge: Polynomial Features ermöglichen es dem Modell, nichtlineare Beziehungen in den Daten zu erfassen, die ein lineares Modell alleine nicht erkennen könnte.
- Verbesserte Modellleistung: Durch die zusätzlichen Merkmale kann das Modell oft genauer werden, besonders wenn die Daten tatsächlich eine nichtlineare Struktur aufweisen.
- Tools und Libraries: Scikit-learn bietet PolynomialFeatures, um solche Merkmale leicht zu generieren

## Nachteile

- Overfitting: Zu viele polynomiale Merkmale können das Modell „überanpassen“, also zu stark auf die Trainingsdaten fixieren, sodass es auf neuen Daten schlecht generalisiert.
- Höherer Rechenaufwand: Die Berechnung und Verarbeitung vieler Polynomial Features kann das Training des Modells verlangsamen und die Anforderungen an Speicher und Rechenleistung erhöhen.
- Anwendungsbereiche: Besonders nützlich bei linearen Modellen für Daten mit nichtlinearen Beziehungen

Der Unterschied zwischen quadratischen und kubischen Termen liegt in der Potenz, auf die ein Merkmal angehoben wird:

- 1. Quadratische Terme:** Hier wird ein Merkmal zum Quadrat genommen, also mit der Potenz 2. Beispiel: Fläche<sup>2</sup> bedeutet, dass der Wert des Merkmals "Fläche" mit sich selbst multipliziert wird (z. B.  $50 \text{ m}^2 * 50 \text{ m}^2 = 2500 \text{ m}^{22}$ ). Quadratische Terme helfen dabei, einfache, gekrümmte Beziehungen darzustellen.
- 2. Kubische Terme:** Hier wird ein Merkmal mit der Potenz 3 angehoben, also "hoch drei" genommen. Beispiel: Fläche<sup>3</sup> bedeutet, dass der Wert der "Fläche" dreimal miteinander multipliziert wird (z. B.  $50 \text{ m}^2 * 50 \text{ m}^2 * 50 \text{ m}^2 = 125000 \text{ m}^3$ ). Kubische Terme sind hilfreich, um noch komplexere und steilere Kurven darzustellen.

### **Zusammengefasst:**

- Quadratische Terme (Fläche<sup>2</sup>) bilden leichte Krümmungen in den Daten ab.
- Kubische Terme (Fläche<sup>3</sup>) ermöglichen die Abbildung stärkerer, nichtlinearer Effekte und können komplexere Muster darstellen.

## Target Encoding

Bei der Target-Kodierung wird für jede Kategorie der durchschnittliche Zielwert berechnet, um so die kategorialen Werte durch numerische Werte zu ersetzen. Dies ist besonders nützlich für Merkmale mit vielen einzigartigen Werten.

Vorteile:

- Verbessert die Vorhersagegenauigkeit, indem die Beziehung zum Zielwert erfasst wird.

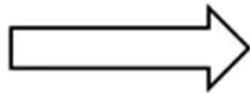
Nachteile:

- Risiko für Overfitting, insbesondere wenn die Kategorien sehr wenige Datenpunkte enthalten.

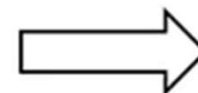
## Beispiel:

### Target Encoding

Feature	Target
Apple	0
Banana	1
Apple	0
Banana	0
Banana	1



Feature	Average(Target)
Apple	0
Banana	$2/3 = 0.66$



Feature	Encoded
Apple	0
Banana	0.66
Apple	0
Banana	0.66
Banana	0.66

## Frequency Encoding:

ersetzt jede Kategorie eines Merkmals durch ihre Häufigkeit im Trainingsdatensatz, wodurch ein numerischer Wert zwischen 0 und 1 entsteht. Diese Methode eignet sich gut, wenn die Häufigkeit einer Kategorie für den Zielwert relevant ist. Neue, unbekannte Kategorien werden automatisch mit 0 kodiert, und der Logarithmus kann helfen, große Häufigkeitsunterschiede auszugleichen.

Height	Sex	Frequency Encoding →	Height	Sex
173.1	Male		173.1	0.4
160.4	Female		160.4	0.6
178.5	Male		178.5	0.4
155.5	Female		155.5	0.6
163.7	Female		163.7	0.6



Vorteile:

- Einfach, leistungsfähig und gut interpretierbar.

Nachteile:

- Unterscheidet nicht zwischen Kategorien gleicher Häufigkeit.
- Kann manchmal die Vorhersagekraft nicht steigern.



# Gleitender Durchschnitt

## Definition und Ziel:

### Definition:

Der gleitende Durchschnitt ist ein Verfahren zur Glättung von Zeitreihendaten, bei dem regelmäßig (z.B. monatlich) der Durchschnitt von Werten über eine festgelegte Periode (z.B. 3 Monate) berechnet wird. Er hilft dabei, Schwankungen zu reduzieren und langfristige Trends sichtbar zu machen.

### Ziel und Vorteile:

#### Rauschunterdrückung:

Der gleitende Durchschnitt glättet rauschhafte Daten und reduziert zufällige Schwankungen, was besonders hilfreich bei Zeitreihendaten ist.

**Effizienz und Einfachheit:** Da es eine recheneffiziente Methode ist, lässt sich der gleitende Durchschnitt schnell berechnen und in Datenvorverarbeitungs-Pipelines integrieren.

**Feature-Engineering:** Er eignet sich zur Generierung von glatten Eingabedaten oder Features, die für bestimmte Modelle hilfreicher sein können als rohe Daten.

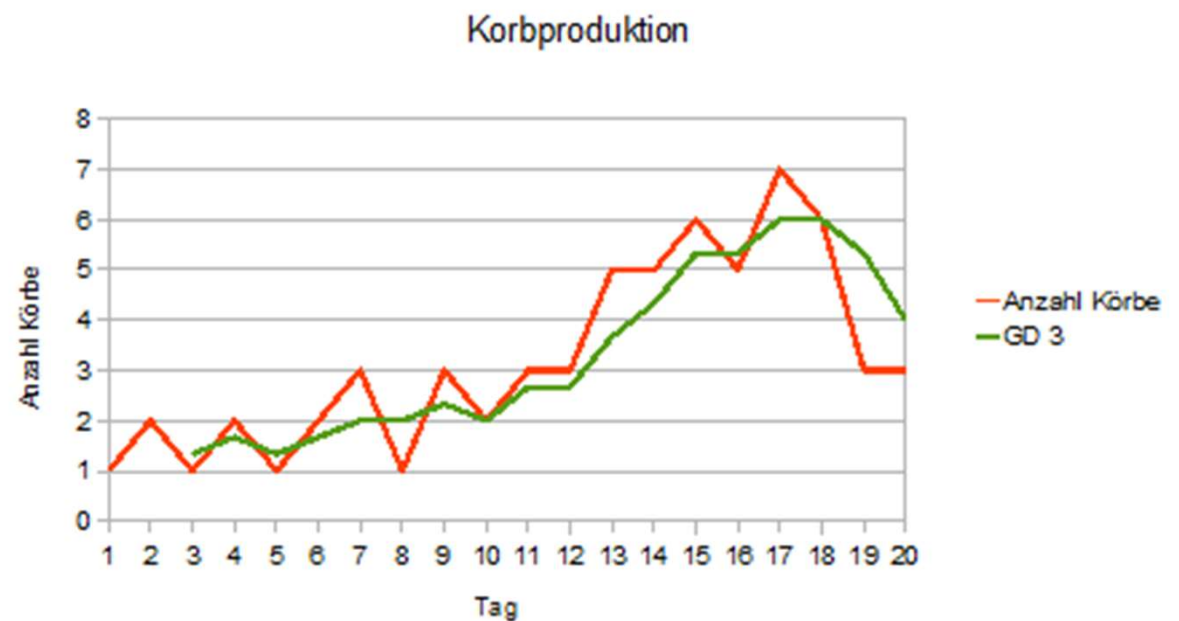
# Wie funktioniert es?

- 1. Datenmenge wählen:.
- 2. Durchschnitt berechnen
- 3. Periode verschieben:
- 4. Wiederholen.
- Varianten :
  - **Einfacher Gleitender Durchschnitt (SMA):** Durchschnitt der letzten n Werte, alle gleich gewichtet. Gut für allgemeine Trends.
  - **Exponentiell Gleitender Durchschnitt (EMA):** Neuere Werte werden stärker gewichtet, reagiert schneller auf Veränderungen. Häufig in der Finanzanalyse genutzt.
  - **Gewichteter Gleitender Durchschnitt (WMA):** Neuere Werte erhalten mehr Gewicht als ältere, aber nicht exponentiell.

Monat	Umsatz (€)	Gleitender Durchschnitt
Januar	1000	-
Februar	2000	-
März	3000	3000
April	4000	3000
Mai	5000	4000

## Beispiel :3-Tage- Gleitender Durchschnitt (GD 3):

- Der grüne GD 3 glättet die täglichen Schwankungen, wodurch der allgemeine Trend klarer wird. Die kurzfristigen Zick-Zack-Bewegungen werden ausgeglichen



# Gleitender Durchschnitt: Herausforderungen und Tools

## Nachteile des Gleitenden Durchschnitts im Machine Learning



### **Begrenzte Vorhersagekraft:**

Basierend nur auf  
historischen Daten;  
keine eigenständige  
Prognose zukünftiger  
Werte.



**Informationsverlust:**  
Glättung kann feine  
Muster oder kurzfristige  
Trends übersehen.



**Verzögerte Reaktion:**  
Langsame Anpassung  
an plötzliche  
Trendwechsel,  
besonders bei längeren  
Perioden.

## Tools und Bibliotheken

- **Pandas:** Für Berechnung und Anwendung gleitender Durchschnitte auf Zeitreihen.
- **scikit-learn:** Bietet integrierte Funktionen zur Datenvorverarbeitung.
- **statsmodels:** Für fortgeschrittene Zeitreihenanalysen und Glättungstechniken.



# Saisonalitätsmerkmale :

## Definition

- Saisonalität bezeichnet vorhersehbare, wiederkehrende Muster und Veränderungen, die innerhalb eines Jahres auftreten.
- Sie basieren auf saisonalen Zyklen, die zu regelmäßigen Schwankungen führen, beispielsweise wöchentlichen, monatlichen oder jährlichen Veränderungen.
- Ursachen sind oft jahreszeitliche, kalendarische oder handelsspezifische Einflüsse.
- Durch Feature Engineering mit Saisonalitätsmerkmalen lassen sich saisonale Muster in Zeitreihen (wie z. B. monatliche Verkaufszahlen) erkennen und Vorhersagen verbessern.
- **Wie funktioniert es?**
- Saisonalitätsmerkmale werden durch Extraktion von Zeitinformationen wie Wochentag, Monat, Quartal oder Jahreszeit erstellt. Diese Werte werden dann als separate Merkmale in das Modell integriert, wodurch saisonale Effekte berücksichtigt werden.



# Vorteile und Herausforderungen im Feature Engineering

## Vorteile und Ziele:

- **Verbesserte Prognosegenauigkeit:** Saisonale Merkmale helfen dabei, wiederkehrende Muster besser zu erfassen, was die Vorhersagegenauigkeit erhöht, insbesondere für periodische Daten.
- **Vereinfachte Interpretation:** Saisonale Effekte ermöglichen Einblicke in zyklische Trends (z.B. Umsatzspitzen in bestimmten Monaten), was zur besseren Entscheidungsfindung beiträgt.

## Nachteile:


- **Komplexere Modellierung:** Saisonale Muster müssen explizit modelliert werden, was die Komplexität erhöht und möglicherweise zusätzliche Features (z.B. Monat, Quartal) erfordert.
- **Benötigte Datenmenge:** Für präzise Saisonmuster sind Daten aus mehreren Zyklen (z.B. Jahren) nötig, was die Datenanforderungen erhöht.

Saisonalitätsmerkmale im Feature Engineering für Zeitreihen  
Beispiel: Monatliche Verkaufszahlen

- **Beispiel-Ausgabe:**
- **Datum:** 2020-01-31 | **Verkäufe:** 100 | **Monat:** 1 | **Jahresquartal:** 1 | **Weihnachtssaison:** 0
- **Datum:** 2020-12-31 | **Verkäufe:** 250 | **Monat:** 12 | **Jahresquartal:** 4 | **Weihnachtssaison:** 1

```
import pandas as pd
data = {
    'Datum': pd.date_range(start='2020-01-01', periods=24, freq='M'),
    'Verkäufe': [100, 120, 130, 150, 160, 170, 110, 100, 90, 130, 200, 250,
                110, 140, 150, 170, 180, 160, 120, 110, 100, 140, 210, 260]
}
df = pd.DataFrame(data)

df['Monat'] = df['Datum'].dt.month          # Monatsnummer
df['Jahresquartal'] = df['Datum'].dt.quarter # Quartal
df['Weihnachtssaison'] = df['Monat'].apply(lambda x: 1 if x == 12 else 0)
# Ausgabe der ersten Zeilen
print(df.head())
```



# Feature- Engineering für saisonale Merkmale: Ansätze und Tools zur Zeitreihenanalyse

## Varianten:

- *Dummy-Codierung*: Wochentage oder Monate werden als separate Kategorien codiert.
- *Trigonometrische Transformation*: Sinus- oder Kosinus-Transformationen für zyklische Merkmale wie Tageszeit oder Monat, um Übergänge zwischen Perioden zu glätten

## Anwendungsbereiche:

Besonders nützlich in der Analyse von Einzelhandelsdaten, Energieverbrauch (z. B. saisonale Heizkosten) oder anderen Bereichen, in denen es regelmäßige, saisonale Muster gibt.


## Tools und Libraries (optional):

**Pandas** – Für die grundlegende Datenmanipulation und das Extrahieren saisonaler Merkmale wie Monat und Quartal, `pip install pandas`

**Statsmodels** – Für statistische Modelle zur Analyse saisonaler Muster, insbesondere SARIMA und Decomposition-Methoden, die saisonale Effekte isolieren können., `pip install statsmodels`.

**Prophet** – Einfache und leistungsstarke Bibliothek für Zeitreihenprognosen mit integrierter Unterstützung für saisonale Effekte (z. B. wöchentliche, monatliche und jährliche Saisonalität).: `pip install prophet`





## *Vergleich von Skalierungstechniken:* **Normalisierung und Standardisierung**

### → **Was ist Skalierung?**

- Skalierung bedeutet, Daten auf eine einheitliche Größenordnung zu bringen.
- Besonders wichtig in Machine Learning, damit alle Merkmale gleich behandelt werden.

### → **Warum ist Skalierung relevant?**

- Viele Algorithmen sind empfindlich gegenüber unterschiedlichen Größen der Daten.
- Ohne Skalierung können manche Merkmale das Modell zu stark beeinflussen, andere werden vernachlässigt.
- Skalierung macht Merkmale vergleichbar und verbessert die Leistung und Genauigkeit des Modells.

# ***Standardisierung (Z-Score-Scaling):***

→ Es ist eine Methode zur Transformation von Daten, bei der der Mittelwert (Durchschnitt) jedes Features auf 0 gesetzt und die Standardabweichung auf 1 skaliert wird.

→ Ziel ist es, die Daten so zu skalieren, dass alle Features die gleiche Skala und Varianz haben, ohne die ursprüngliche Verteilung zu verändern.

→ Funktionsweise:

z-Wert berechnen mit der Formel:

$$z = \frac{(x - \mu)}{\sigma}$$

- $x$  = zu standardisierender Wert
- $\mu$  = Mittelwert
- $\sigma$  = Standardabweichung

# Beispiel: Vorhersage von Hauspreisen

## → Probleme:

- Hausgröße (100 bis 300 m<sup>2</sup>) hat größere Zahlenwerte als das Alter des Hauses (5 bis 30 Jahre).
- Risiko, dass das Modell sich stärker auf Hausgröße konzentriert, weil die Werte numerisch höher sind.

Haus-Nr	Haus Größe	Alter des Hauses	Preis (in Tausend \$)
1	100	10	250
2	150	20	350
3	200	5	500
4	250	30	600
5	300	15	750

**Originalwerte der Merkmale und Preise**

## → Vorteile der Standardisierung:

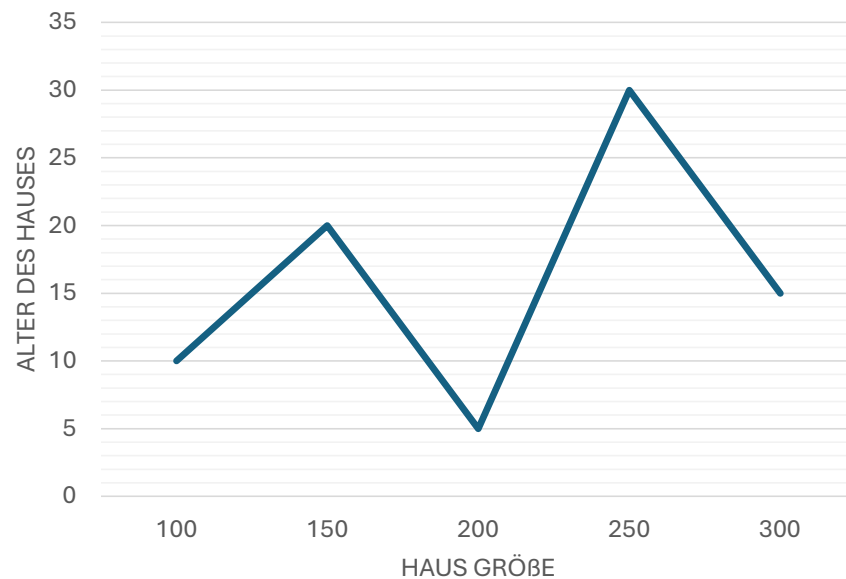
- Bringt beide Merkmale auf dieselbe Skala (Mittelwert 0, Standardabweichung 1)
- Ermöglicht dem Modell, beide Merkmale gleichermaßen zu berücksichtigen
- Verhindert, dass das Modell nur von der größeren Hausgröße beeinflusst wird.
- Ermöglicht, den Einfluss des Alters des Hauses korrekt zu berücksichtigen.

Haus-Nr	Haus Größe	Alter des Hauses	Preis (in Tausend \$)
1	-1,41	-0,68	250
2	-0,71	0,45	350
3	0	-1,24	500
4	0,71	1,58	600
5	1,41	-0,11	750

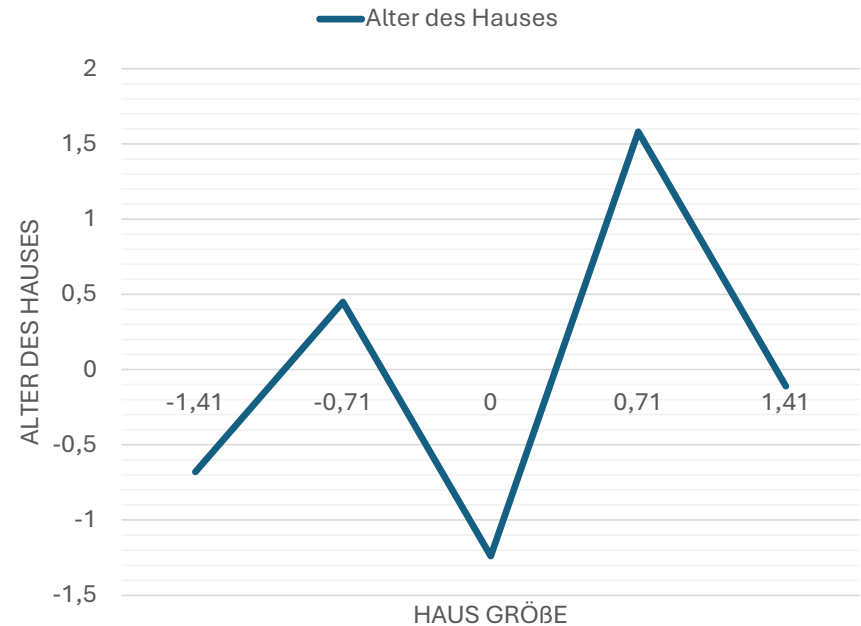
**Standardisierte Werte der Merkmale und Preise**

# Vergleich: *Original- und Standardisierte Merkmale*

Originalwerte der Merkmale und Preise



Standardisierte Werte der Merkmale



# Normalisierung (Min-Max-Scaling):

- Bei der Min-Max-Normalisierung geht es darum, die Werte in einem bestimmten Bereich wie [0, 1] oder [-1, 1] zu skalieren. Man nutzt diese Methode, um die Daten leichter vergleichbar zu machen, besonders wenn die Werte ursprünglich sehr unterschiedlich sind.
- Normalisierte Daten ermöglichen es dem Modell, alle Merkmale unabhängig von deren ursprünglicher Größenordnung zu berücksichtigen.
- Führt zu höherer Genauigkeit und reduziert die Empfindlichkeit gegenüber großen Werten.
- Verhindert, dass einzelne Merkmale das Modell dominieren und sorgt für konsistente Anpassung.
- Funktionsweise:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

X ist der Originalwert.

X' ist der normalisierte Wert.

Xmin ist der kleinste Wert in den Daten.

Xmax ist der größte Wert in den Daten.

# Beispiel: Min-Max-Normalisierung

## → Vorteile:

- **Vergleichbare Skala:** Gehalt und Alter sind auf denselben Bereich gebracht (0 bis 1), was die Daten vergleichbar macht.
- **Bessere Leistung in Algorithmen:** Normalisierte Daten verbessern die Effizienz von maschinellen Lernmodellen, die auf Abstandsberechnungen basieren (z.B. KNN).

Person	Gehalt	Alter
A	60	25
B	80	40
C	50	30
D	90	60
E	100	50

*Datensatz vor der Normalisierung*

## → Nachteile:

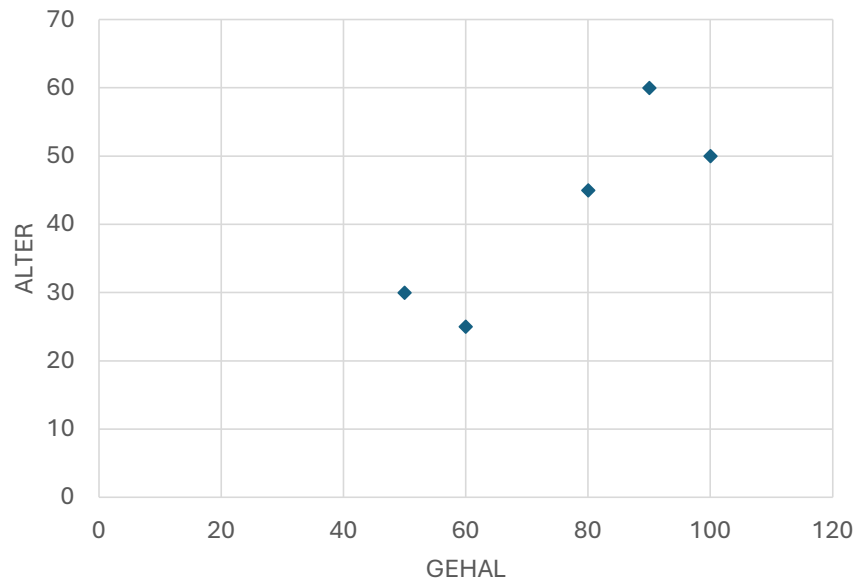
- **Anfällig für Ausreißer:** Extreme Werte (Ausreißer) können den Bereich  $[0, 1]$  verzerren, was die Aussagekraft der Daten verringern kann.
- **Abhängigkeit von den Min/Max-Werten:** Bei neuen oder veränderten Daten muss die Skalierung neu berechnet werden, da sich die Min- und Max-Werte ändern können.

Person	Gehalt	Alter
A	0,25	0,111
B	0,5	0,556
C	0,125	0,222
D	0,625	0,889
E	0,75	0,667

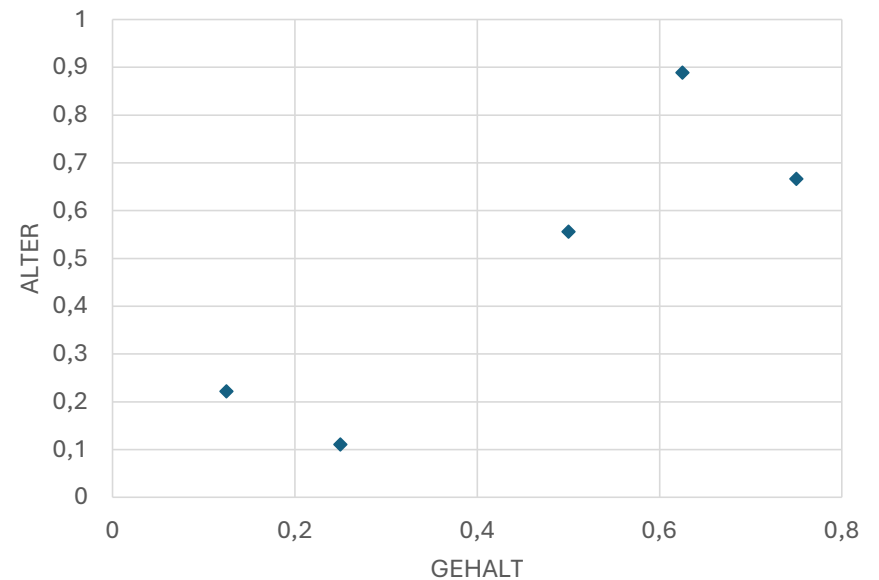
*Normalisierte Daten (Bereich  $[0, 1]$ )*

# ***Einfluss der Normalisierung auf Daten: Vorher und Nachher***

**DATENSATZ VOR DER  
NORMALISIERUNG**



**NORMALISIERTE DATEN  
(BEREICH [0, 1])**



# Anwendungsbereiche:

**Normalisierung (Min-Max-Scaling)** nützlich bei:

1. *Neuronale Netze*: Eingabewerte werden in den gleichen Bereich (z. B. 0 bis 1) gebracht, damit das Training stabiler wird.
2. *Zeitreihen*: Vergleichbarkeit von Daten wie Temperaturen oder Preisen, die oft stark schwanken.
3. *Bildverarbeitung*: Pixelwerte liegen häufig zwischen 0 und 1, was für Modelle einfacher zu verarbeiten ist.

☐ Tools und Libraries:

- I. *scikit-learn*: `MinMaxScaler` schnelle und zuverlässige Normalisierung
- II. *Pandas, NumPy*: `.min()` und `.max()` für eine einfache Berechnung der Skala.

**Standardisierung (Z-Score-Scaling)** nützlich bei:

1. *Linearen Modellen*: Algorithmen wie lineare und logistische Regression sind stabiler mit z-standardisierten Daten.
2. *Distanzbasierten Algorithmen*: Bei Verfahren wie k-Nearest Neighbors und k-Means, die auf Abständen basieren, ist eine vergleichbare Skala der Features wichtig.
3. *Normalverteilten Daten*: Z-Score-Scaling bringt die Daten in eine Verteilung mit Mittelwert 0 und Standardabweichung 1.

☐ Tools und Libraries:

- I. *scikit-learn*: `StandardScaler` berechnet Mittelwert und Standardabweichung automatisch.
- II. *Pandas und NumPy*: `.mean()` und `.std()` für Standardisierung.



## Quellen:

- [www.researchgate.net/figure/Example-of-frequency-encoding\\_fig1\\_364144236](https://www.researchgate.net/figure/Example-of-frequency-encoding_fig1_364144236)  
[Categorical Data Encoding Techniques | by Krishnakanth Naik Jarapala | AI Skunks | Medium](#)  
[Feature Engineering A-Z | Frequency Encoding – Feature Engineering A-Z](#)
- <https://aktien-mit-strategie.de/exponentiell-gleitender-durchschnitt/>