



# Lineare Regression

## Inhalt

- [Lineare Regression](#)
  - [Voraussetzungen](#)
  - [Vorgehen](#)
  - [Berechnung mit SciPy](#)

Hast du mit der [Korrelationsanalyse](#) einen linearen Zusammenhang zwischen zwei Variablen festgestellt, kannst du im Streudiagramm eine Gerade einzeichnen, die möglichst nah durch deine Messpunkte verläuft, wie wir es in der vorigen Einheit schon als optische Hilfestellung getan haben.

## Voraussetzungen

Um das Modell der linearen Regression benutzen zu können, muss auf folgendes geachtet werden:

- Die Werte müssen metrisches Skalenniveau haben ([hier](#) erklärt).
- Es muss ein linearer Zusammenhang zwischen beiden Variablen bestehen ([hier](#) erklärt).
- Ausreißer entfernen: Ausreißer können Lage der Regressionsgeraden ändern. Identifizierst du solche bei der Überprüfung des Streudiagramms hinter der linearen Regression, musst du sie ggf. von der Regressionsrechnung entfernen.
- Die Streuung der Fehlerterme sollte entlang der Regressionsgeraden konstant sein.

## Vorgehen

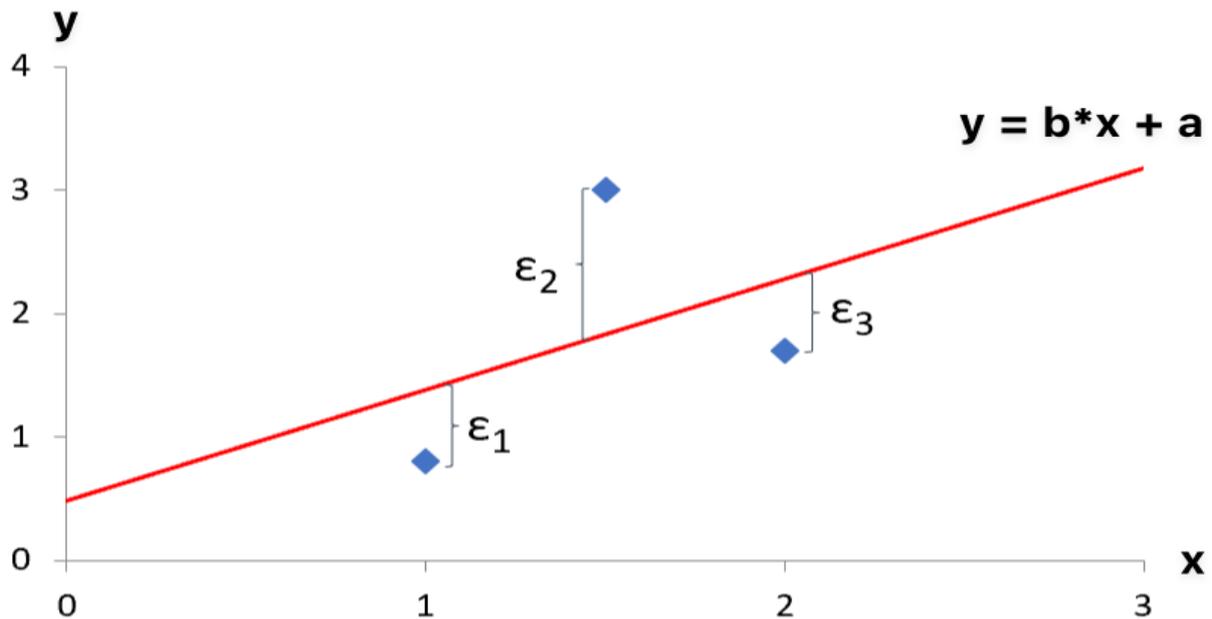
Wir wollen nun für diese Gerade eine Geradengleichung in der Form  $y = b \cdot x + a$  angeben.

- $b$  ist die Steigung (engl. slope) der Geraden.
- $a$  ist der y-Achsenabschnitt (engl. intercept) der Geraden.

Die beiden Werte  $a$  und  $b$  sollen aus den Messwerten so bestimmt werden, dass die Summe aller quadrierten Abweichungen (die sogenannten Fehlerquadrate) in y-



Richtung so klein wie möglich ist.



Es handelt sich hierbei um eine Linie durch 3 Punkte

Kurz: Finde  $a$  und  $b$  so, dass  $\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2$  minimal wird.

Das Ergebnis ist eine einfache lineare Funktion, die sogenannte Regressionsgerade, mit der du zu jedem vorgegebenen  $x$ -Wert den zu erwartenden  $y$ -Wert vorhersagen kannst.

Dieses Verfahren nennt man lineare Regression. Sie ist eine der wichtigsten Verfahren des Maschinellen Lernens. Du kannst hiermit nicht nur lineare Abhängigkeiten zwischen zwei Variablen beschreiben, sondern aus einem bekannten  $x$ -Wert auch Prognosen für einen zu erwartenden  $y$ -Wert erstellen.

## Berechnung mit SciPy

Du musst diese Rechnung nicht selbst durchführen. Python liefert uns auch hierfür wieder fertige Funktionen. Für die lineare Regression benutzen wir aus dem Paket SciPy die Funktion `results = stats.linregress()`, die wir oben schon zur Konstruktion der Hilfsgeraden benutzt haben. Der Rückgabewert `results` ist ein Datenobjekt, das folgende Werte enthält:



- `results.slope`: Steigung  $m$  der Regressionsgeraden
- `results.intercept`: Achsenabschnitt  $a$  der Regressionsgeraden
- `results.rvalue`: Pearson'scher Korrelationskoeffizient  $r$
- `results.pvalue`:  $p$ -Wert-Wert des Signifikanztests
- `results.stderr`: Standardfehler der Steigung (geschätzte Ungenauigkeit von  $m$ )
- `results.intercept_stderr`: geschätzte Ungenauigkeit von  $a$

Da wir bei einer linearen Regression ohnehin eine Korrelationsanalyse und einen Signifikanztest durchführen müssen, liefert die Funktion `linregress()` die entsprechenden Werte  $r$  und  $p$  also praktischerweise gleich mit.

In einem vorangegangenen Video haben wir einen [Immobilienpreisrechner](#) gezeigt. In diesem Berechnungsmodell ergibt sich der Immobilienpreis  $P$  aus einem Anteil, der proportional zur Wohnfläche  $W$  ist plus weiteren Anteilen, so dass der Preis in der Form  $P = a + b \cdot W$  geschrieben werden kann. Dabei haben wir die Faktoren  $a$  und  $b$  als bekannt angenommen. Aber woher bekommen wir die Werte von  $a$  und  $b$ ? Wir erhalten sie, indem wir anhand einer Testdatenbank eine lineare Regression durchführen und so Durchschnittswerte für  $a$  und  $b$  ermitteln. Die Testdatenbank sollte eine große Liste von Wohnungen enthalten, die sich möglichst nur durch die Wohnfläche  $W$  und die zugehörigen Preise  $P$  unterscheiden.

Führen wir nun an folgenden Testdaten für Immobilienpreise eine lineare Regression durch. Die Testdaten geben die Wohnfläche in  $m^2$  und die Preise in tausend Euro an.

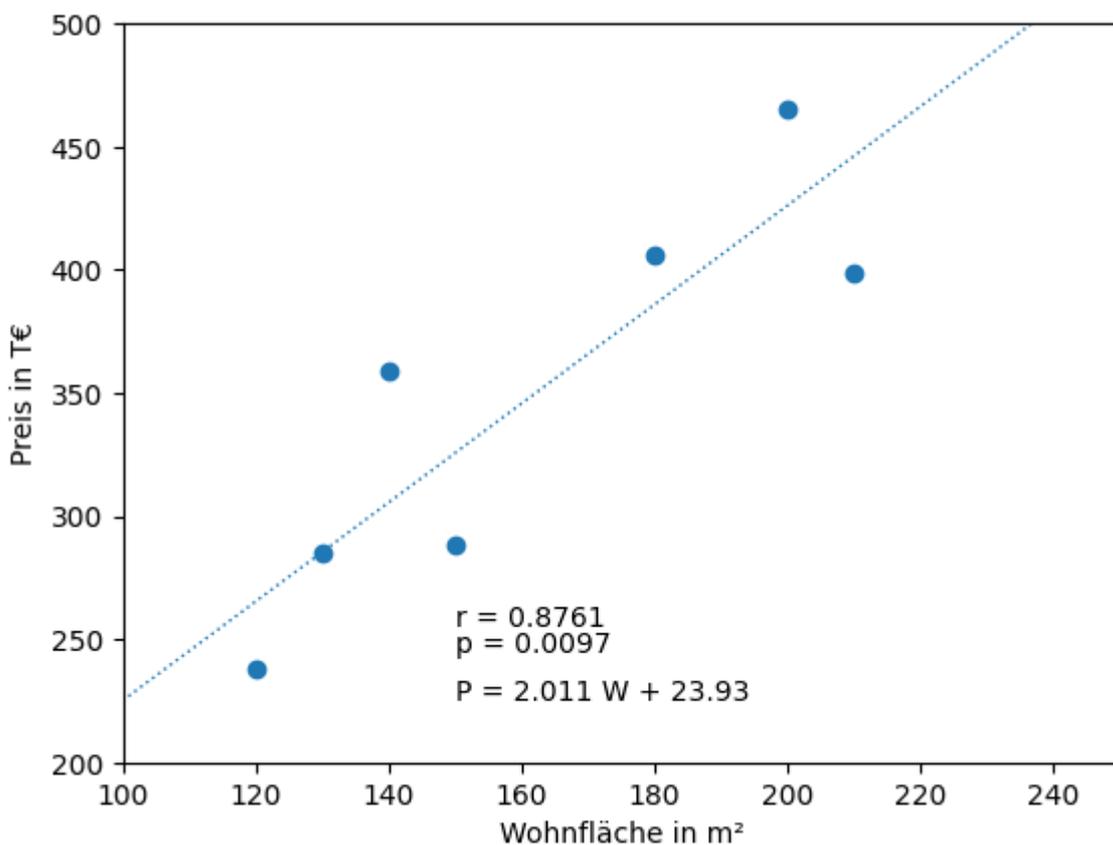
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats

df = pd.DataFrame([(120,238), (150,288), (180,406), (200,465),
(210,399), (140,359), (130,285)], columns=["Wohnfläche", "Preis"])
df.dropna(inplace=True)
```



```
# Linie berechnen
def linie(x):
    return b * x + a
results = stats.linregress(df["Wohnfläche"],df["Preis"])
r = results.rvalue
p = results.pvalue
b = results.slope
a = results.intercept
y = list(map(linie, [100,250]))

# Scatterplot zeichnen
plt.scatter(df["Wohnfläche"],df["Preis"]) # Punkte zeichnen
plt.axis([100,250,200,500]) # Achsenskalierung
plt.xlabel("Wohnfläche in m²") # Achsenbeschriftungen
plt.ylabel("Preis in T€")
plt.plot([100,250], y, linewidth=1, linestyle=":") # Linie
plt.text(150,255,"r = "+str(r.round(4))) # r anzeigen
plt.text(150,245,"p = "+str(p.round(4))) # p anzeigen
plt.text(150,225,"P = "+str(b.round(3))+ " W + "+str(a.round(2)))
plt.show()
```



Wir erhalten aus unseren Testdaten einen Schätzwert für den Parameter  $m_j$   $\begin{equation} b \end{equation}$   $m_j$ , der mit 2011 Euro pro m² Wohnfläche schon sehr



nah an dem Wert liegt, den wir für unseren Immobilienpreisrechner verwendet haben.

Der Achsenabschnittswert, den man für eine Wohnfläche von 0 m<sup>2</sup> erhält, von 23.930 Euro und die relativ starke Streuung der Messpunkte lassen jedoch vermuten, dass es neben der Wohnfläche noch weitere Einflussgrößen auf den Preis gibt. Tatsächlich haben wir gesehen, dass u.a. die Grundstücksfläche und das Alter der Immobilie einen Einfluss auf den Preis haben. Wenn wir diese Größen in unsere Testdaten mit aufnehmen, können wir jeweils eine weitere lineare Regression des Gesamtpreises  $P$  bezüglich der Grundstücksfläche  $G$  beziehungsweise des Alters  $A$  durchführen und so die jeweiligen Parameterwerte  $c$ ,  $d$  usw. bestimmen.

Der Gesamtpreis setzt sich dann aus der Summe der einzelnen Beiträge zusammen:

$$P =$$

$$\mathbf{P} = \mathbf{a} + \mathbf{bW} + \mathbf{cG} + \mathbf{dA} + \dots$$