

# Blatt 7

Vincent Kümmerle und Elvis Gnaglo

3. Dezember 2025

## 1 Zotero

Machinelles Lernen wird verwendet, um Muster in Datensammlungen herauszufinden. [1]

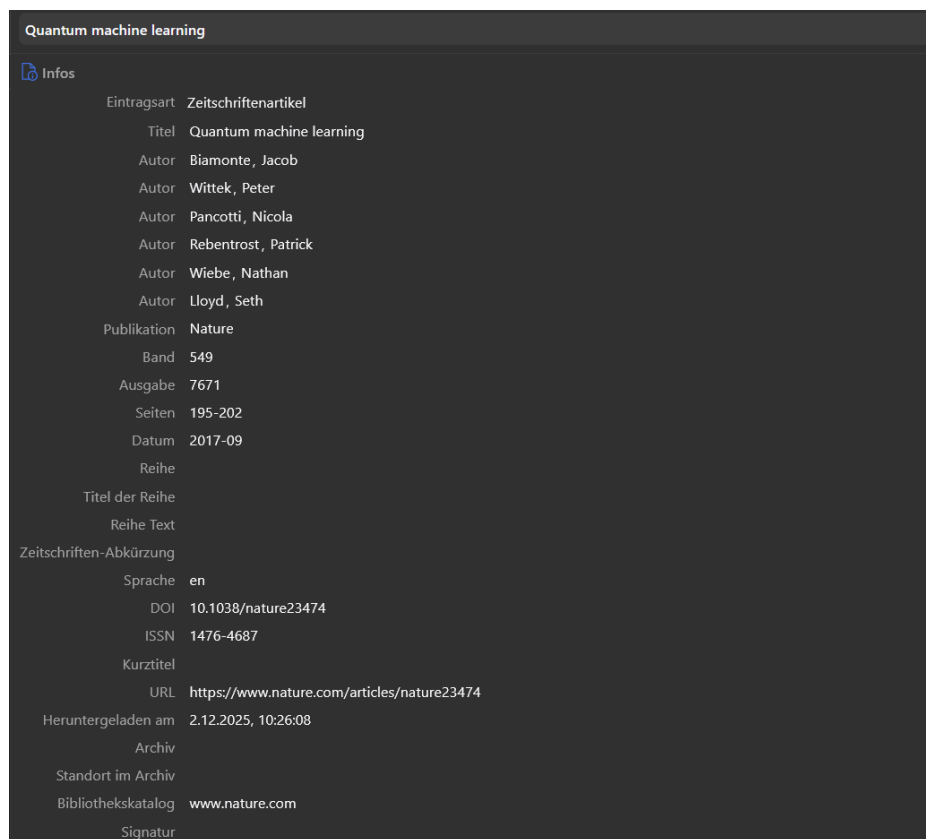


Abbildung 1: Zeitschriftenartikel über Quantum Machine Learning in der Zotero-Bibliothek.

Das Buch "Experimentalphysik 1" von Wolfgang Demtröder erklärt die Grundlagen der Mechanik und Wärmelehre. [2]

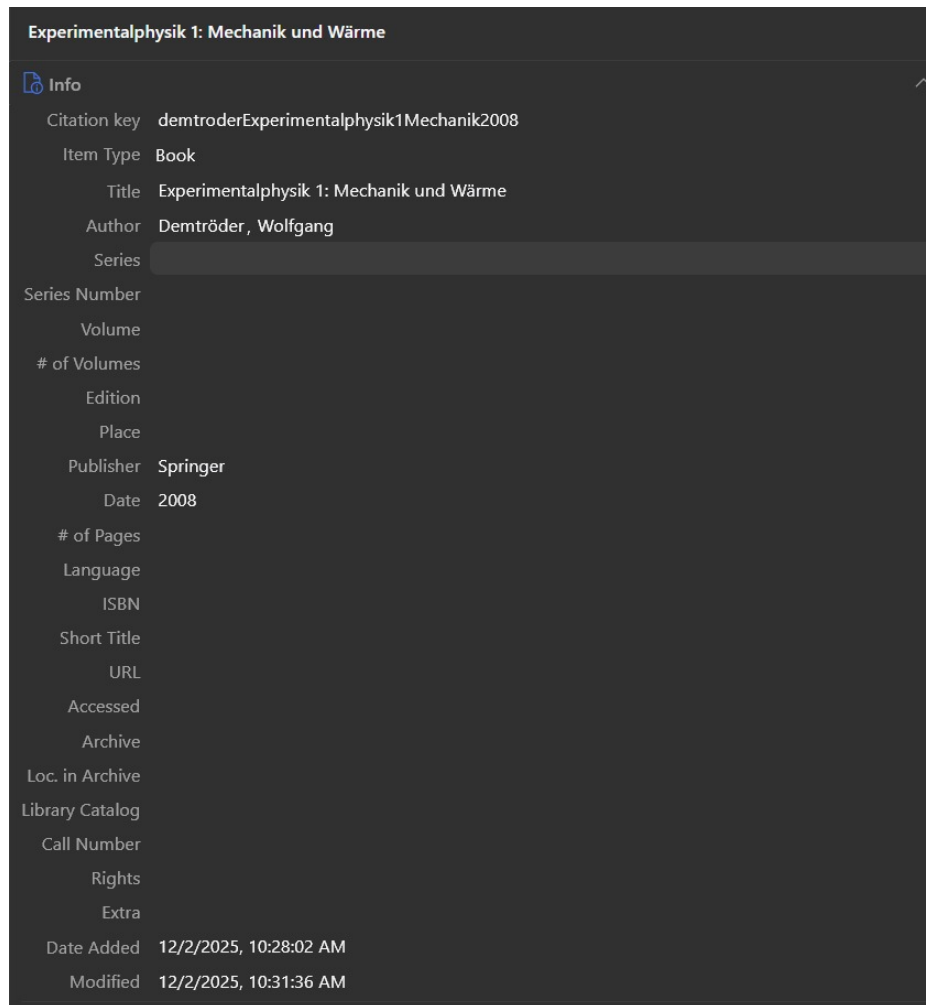


Abbildung 2: Buch über die Mechanik und Wärmelehre in der Physik.

## 2 List Comprehension

```
1 import random
2 import matplotlib.pyplot as plt
3 import math
4
5 def generate_random_point_in_square(length):
6     """
7     Generate a random point in a square of a given side length
8     centered around the origin with the sides aligned with x- and
9     y-axis.
10    """
11    return tuple([length*random.uniform(-0.5, 0.5), length*random.
12                  uniform(-0.5, 0.5)])
13
14 def distance(p1, p2):
```

```

12     """
13     Measures the Euclidean distance between the points p1 and p2 in 2
        D.
14     """
15     # Berechnung des euklidischen Abstands: wurzel((x2-x1)^2 + (y2-y1
        )^2)
16     return math.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)
17
18 def point_in_circle(p, center=(0,0), radius=1.0):
19     """
20     Checks if the provided point p lies in a circle of a given radius
        around the provided center.
21     """
22     # Ein Punkt liegt im Kreis, wenn sein Abstand zum Mittelpunkt
        kleiner oder gleich dem Radius ist.
23     return distance(p, center) <= radius
24
25
26 # TODO: generate a list of 10000 random points in a square of side
        length 2.0 using generate_random_point_in_square and list
        comprehension
27 # Wir nutzen _ als Laufvariable, da wir den Index nicht benötigen.
28 full_list_of_points = [generate_random_point_in_square(2.0) for _ in
        range(10000)]
29
30 # TODO: generate a list of those points in full_list_of_points that
        are within a radius of 1.0 of the origin (again using list
        comprehension)
31 # Wir filtern die Liste basierend auf der Bedingung point_in_circle
32 filtered_list_of_points = [p for p in full_list_of_points if
        point_in_circle(p, center=(0,0), radius=1.0)]
33
34 # TODO: print the ratio of the number of points in
        filtered_list_of_points to the total number of generated points
35 ratio = len(filtered_list_of_points) / len(full_list_of_points)
36 print(f"Verhältnis (Punkte im Kreis / Gesamt): {ratio}")
37 print(f"Annäherung an Pi (Verhältnis * 4): {ratio * 4}")
38 # Erklärung: Fläche Kreis / Fläche Quadrat = (pi*r^2) / (2r)^2 = pi
        /4. Daher ist das Verhältnis ca. pi/4.
39
40 # TODO: Generate separate lists of the x- and y-values of
        full_list_of_points and filtered_list_of_points
41 # x-Werte sind am Index 0, y-Werte am Index 1 jedes Tupels
42 x_values_full = [p[0] for p in full_list_of_points]
43 y_values_full = [p[1] for p in full_list_of_points]

```

```

44
45 x_values_filtered = [p[0] for p in filtered_list_of_points]
46 y_values_filtered = [p[1] for p in filtered_list_of_points]
47
48 # Plotten
49 plt.figure(figsize=(6, 6)) # Quadratische Darstellung für korrekte
    Proportionen
50 plt.scatter(x_values_full, y_values_full, s=1, label='Außerhalb',
    color='blue')
51 plt.scatter(x_values_filtered, y_values_filtered, s=1, label='
    Innerhalb', color='red')
52 plt.legend(loc='upper right')
53 plt.title(f'Monte Carlo Simulation (Verhältnis = {ratio})')
54 plt.axis('equal') # Wichtig, damit der Kreis rund und nicht verzerrt
    aussieht
55 plt.show()

```

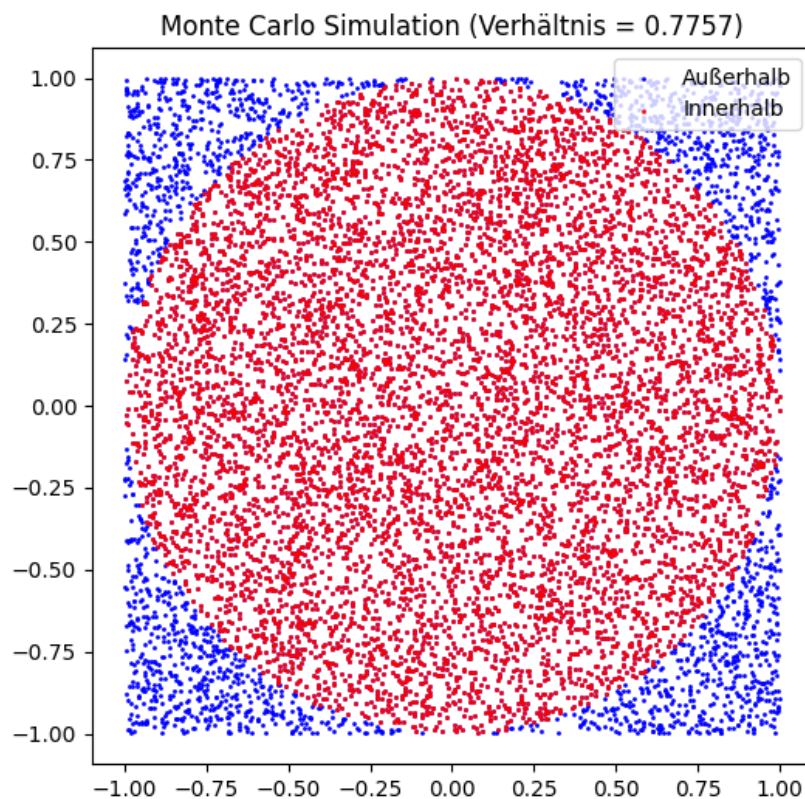


Abbildung 3: Monte Carlo Simulation

### 3 Dateien lesen und Daten plotten

---

#### Literaturverzeichnis

- [1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, *Nature* **2017**, 549, 195–202.
- [2] W. Demtröder, *Experimentalphysik 1: Mechanik Und Wärme*, Springer, **2008**.