Blatt 7

Vincent Kümmerle und Elvis Gnaglo

4. Dezember 2025

1 Zotero

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

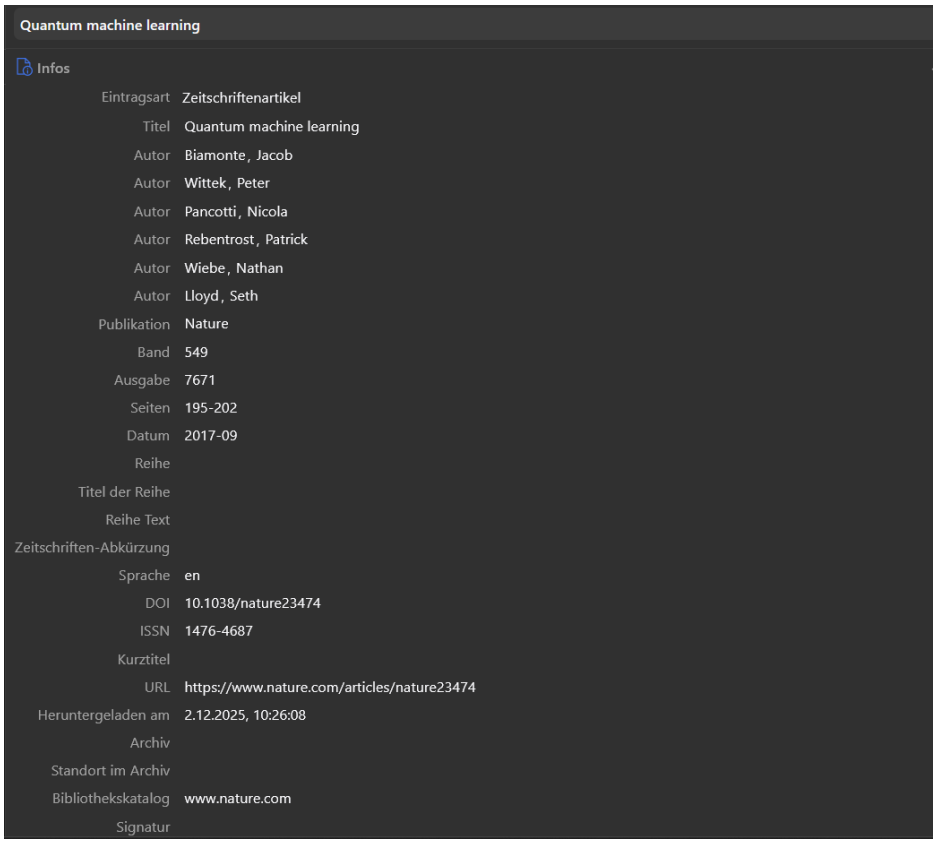

Abbildung 1: Zeitschriftenartikel über Quantum Machine Learning in der Zotero-Bibliothek.

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

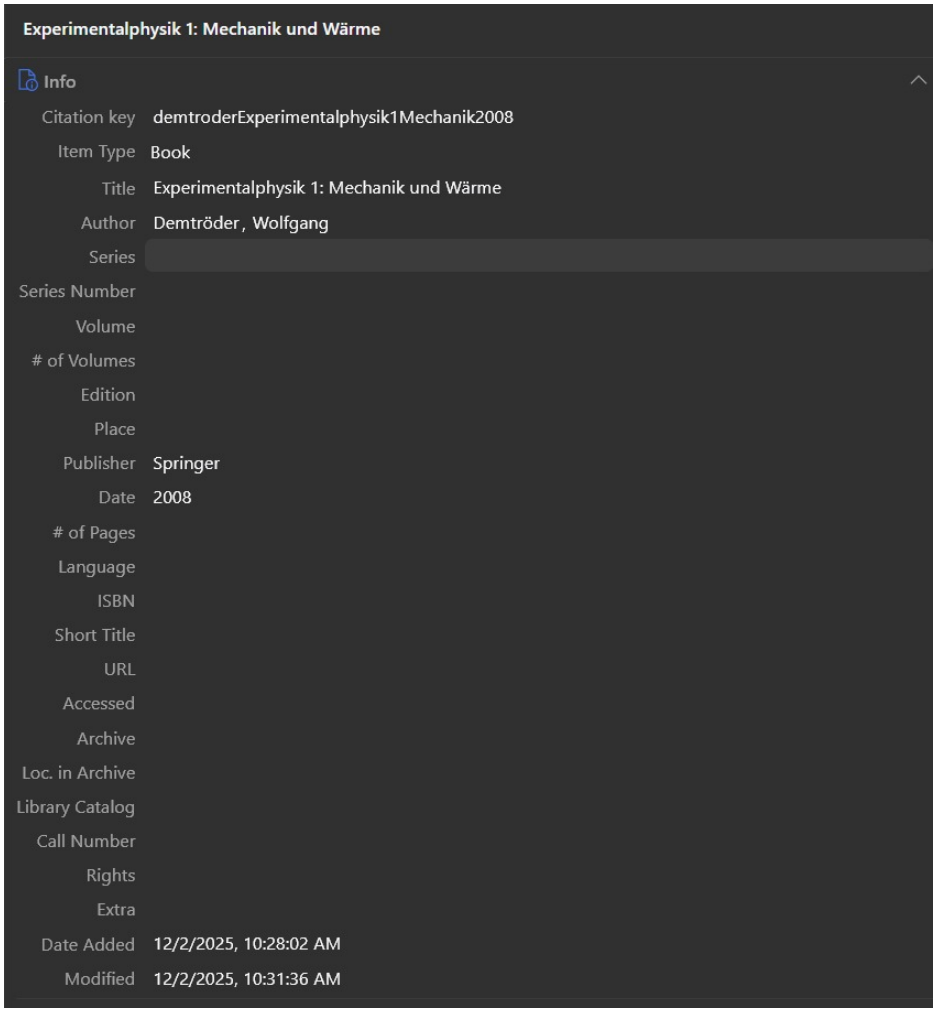

Abbildung 2: Buch über die Mechanik und Wärmelehre in der Physik.

2 List Comprehension

```python
import random
import matplotlib.pyplot as plt
import math

def generate_random_point_in_square(length):
    """
    Generate a random point in a square of a given side length
        centered around the origin with the sides aligned with x- and
        y-axis.
    """
    return tuple([length*random.uniform(-0.5, 0.5), length*random.
        uniform(-0.5, 0.5)])

def distance(p1, p2):
```

```
    """
    Measures the Euclidean distance between the points p1 and p2 in 2 D.
    """
    # TODO: wurzel((x2-x1)^2 + (y2-y1)^2)
    return math.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)

def point_in_circle(p, center=(0,0), radius=1.0):
    """
    Checks if the provided point p lies in a circle of a given radius around the provided center.
    """
    # TODO: Abstand zum Mittelpunkt kleiner/gleich dem Radius
    return distance(p, center) <= radius


# 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
# Liste erzeugen mit _ als Laufvariable, kein Index nötig
full_list_of_points = [generate_random_point_in_square(2.0) for _ in range(10000)]

# 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)
# Liste filtern basierend auf der Bedingung point_in_circle
filtered_list_of_points = [p for p in full_list_of_points if point_in_circle(p, center=(0,0), radius=1.0)]

# TODO: print the ratio of the number of points in filtered_list_of_points to the total number of generated points
ratio = len(filtered_list_of_points) / len(full_list_of_points)
print(f"Verhältnis (Punkte im Kreis / alle Punkte): {ratio}")
print(f"Annäherung an Pi (Verhältnis * 4): {ratio * 4}")
# Verhältnis Punkte im Kreis (in gefilterter Liste) / alle Punkte = Fläche Kreis / Fläche Quadrat = (pi*r^2) / (2r)^2 = pi/4
# Somit konvergiert es für n gegen Unendlich gegen pi/4.

# TODO: Generate separate lists of the x- and y-values of full_list_of_points and filtered_list_of_points
# x-Werte sind am Index 0, y-Werte am Index 1 jedes Tupels
x_values_full = [p[0] for p in full_list_of_points]
y_values_full = [p[1] for p in full_list_of_points]

```

```
x_values_filtered = [p[0] for p in filtered_list_of_points]
y_values_filtered = [p[1] for p in filtered_list_of_points]

# Plotten
plt.figure(figsize=(6, 6)) # Quadratische Darstellung für korrekte Proportionen
plt.scatter(x_values_full, y_values_full, s=1, label='außerhalb', color='blue')
plt.scatter(x_values_filtered, y_values_filtered, s=1, label='innerhalb', color='red')
plt.legend(loc='upper right')
plt.title(f'Monte Carlo Simulation (Verhältnis = {ratio})')
plt.axis('equal') # damit der Kreis nicht verzerrt aussieht
plt.show()
```

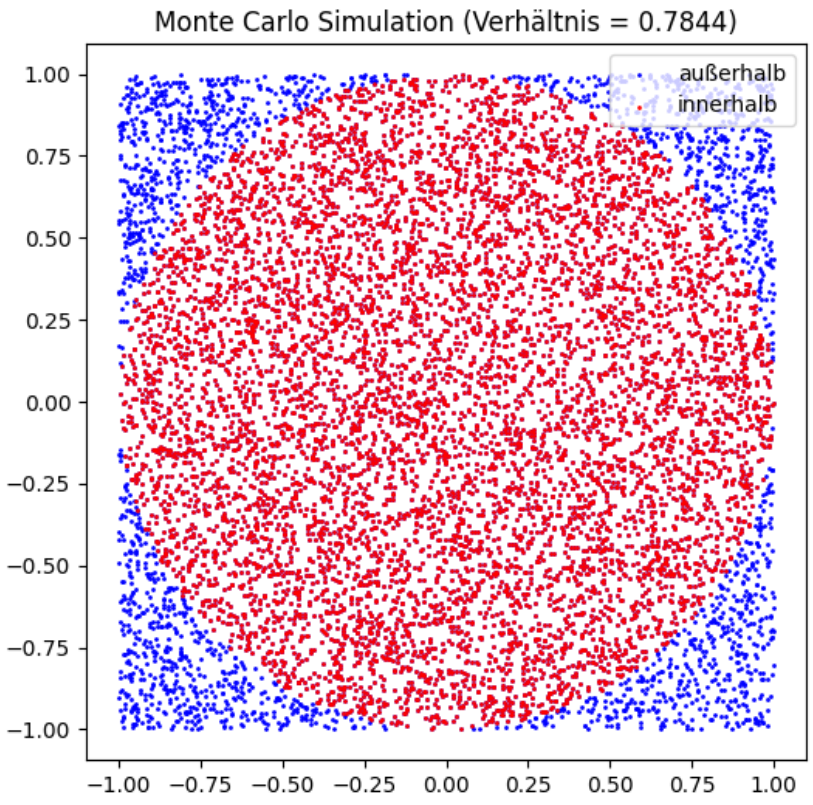

Abbildung 3: Monte Carlo Simulation

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**.