

Blatt 7

Vincent Kümmerle und Elvis Gnaglo

3. Dezember 2025

1 Zotero

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

The screenshot shows a Zotero library entry for a scientific article. The title is "Quantum machine learning". The entry details are as follows:

Infos

Eintragsart: Zeitschriftenartikel

Titel: Quantum machine learning

Autor: Biamonte, Jacob

Autor: Wittek, Peter

Autor: Pancotti, Nicola

Autor: Rebentrost, Patrick

Autor: Wiebe, Nathan

Autor: Lloyd, Seth

Publikation: Nature

Band: 549

Ausgabe: 7671

Seiten: 195-202

Datum: 2017-09

Reihe

Titel der Reihe

Reihe Text

Zeitschriften-Abkürzung

Sprache: en

DOI: 10.1038/nature23474

ISSN: 1476-4687

Kurztitel

URL: <https://www.nature.com/articles/nature23474>

Heruntergeladen am: 2.12.2025, 10:26:08

Archiv

Standort im Archiv

Bibliothekskatalog: www.nature.com

Signatur

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]

Experimentalphysik 1: Mechanik und Wärme

 Info ^

Citation key demtroderExperimentalphysik1Mechanik2008

Item Type Book

Title Experimentalphysik 1: Mechanik und Wärme

Author Demtröder, Wolfgang

Series

Series Number

Volume

of Volumes

Edition

Place

Publisher Springer

Date 2008

of Pages

Language

ISBN

Short Title

URL

Accessed

Archive

Loc. in Archive

Library Catalog

Call Number

Rights

Extra

Date Added 12/2/2025, 10:28:02 AM

Modified 12/2/2025, 10:31:36 AM

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):
15     """
16         Measures the Euclidean distance between the points p1 and p2 in 2
17         D.
18     """
19     # Berechnung des euklidischen Abstands: wurzel((x2-x1)^2 + (y2-y1
20     )^2)
21     return math.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)
22
23 def point_in_circle(p, center=(0,0), radius=1.0):
24     """
25         Checks if the provided point p lies in a circle of a given radius
26         around the provided center.
27     """
28     # Ein Punkt liegt im Kreis, wenn sein Abstand zum Mittelpunkt
29         kleiner oder gleich dem Radius ist.
30     return distance(p, center) <= radius
31
32
33 # TODO: generate a list of 10000 random points in a square of side
34     length 2.0 using generate_random_point_in_square and list
35     comprehension
36 # Wir nutzen _ als Laufvariable, da wir den Index nicht benötigen.
37 full_list_of_points = [generate_random_point_in_square(2.0) for _ in
38     range(10000)]
39
40 # TODO: generate a list of those points in full_list_of_points that
41     are within a radius of 1.0 of the origin (again using list
42     comprehension)
43 # Wir filtern die Liste basierend auf der Bedingung point_in_circle
44 filtered_list_of_points = [p for p in full_list_of_points if
45     point_in_circle(p, center=(0,0), radius=1.0)]
46
47 # TODO: print the ratio of the number of points in
48     filtered_list_of_points to the total number of generated points
49 ratio = len(filtered_list_of_points) / len(full_list_of_points)
50 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
39           /4. Daher ist das Verhältnis ca. pi/4.
40
41 # TODO: Generate separate lists of the x- and y-values of
42   full_list_of_points and filtered_list_of_points
43 # x-Werte sind am Index 0, y-Werte am Index 1 jedes Tupels
44 x_values_full = [p[0] for p in full_list_of_points]
45 y_values_full = [p[1] for p in full_list_of_points]
46
47 x_values_filtered = [p[0] for p in filtered_list_of_points]
48 y_values_filtered = [p[1] for p in filtered_list_of_points]
49
50 # Plotten
51 plt.figure(figsize=(6, 6)) # Quadratische Darstellung für korrekte
52   Proportionen
53 plt.scatter(x_values_full, y_values_full, s=1, label='Außerhalb',
54   color='blue')
55 plt.scatter(x_values_filtered, y_values_filtered, s=1, label='
56   Innerhalb', color='red')
57 plt.legend(loc='upper right')
58 plt.title(f'Monte Carlo Simulation (Verhältnis = {ratio})')
59 plt.axis('equal') # Wichtig, damit der Kreis rund und nicht verzerrt
60   aussieht
61 plt.show()

```

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**.