



# Einsatz von Geodaten im Marketing

Mietpreisfaktoren Zürich

Gruppe 2

Anja Kovanovic | Denis Machacka | Ranujan Kumar



# Inhaltsverzeichnis

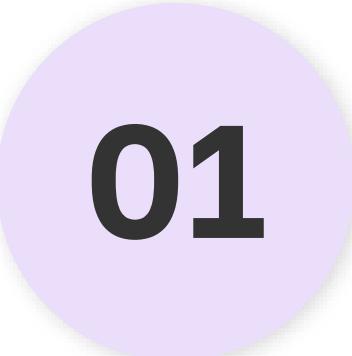


**01** Projekteinleitung

**02** Scraping

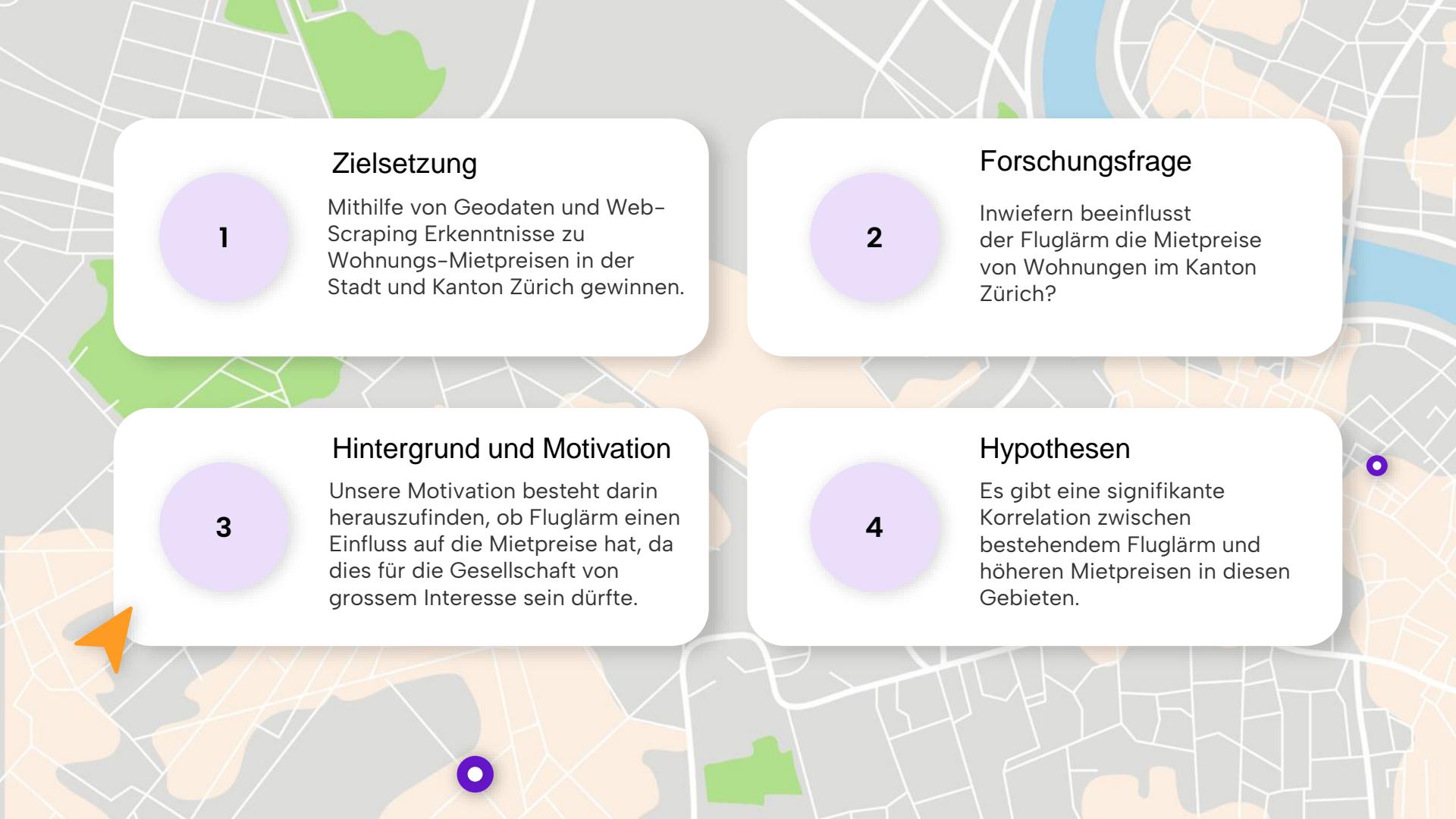
**03** QGIS

**04** Auswertung und Fazit



**01**

# Projekteinleitung



1

## Zielsetzung

Mithilfe von Geodaten und Web-Scraping Erkenntnisse zu Wohnungs-Mietpreisen in der Stadt und Kanton Zürich gewinnen.

2

## Forschungsfrage

Inwiefern beeinflusst der Fluglärm die Mietpreise von Wohnungen im Kanton Zürich?



3

## Hintergrund und Motivation

Unsere Motivation besteht darin herauszufinden, ob Fluglärm einen Einfluss auf die Mietpreise hat, da dies für die Gesellschaft von grossem Interesse sein dürfte.

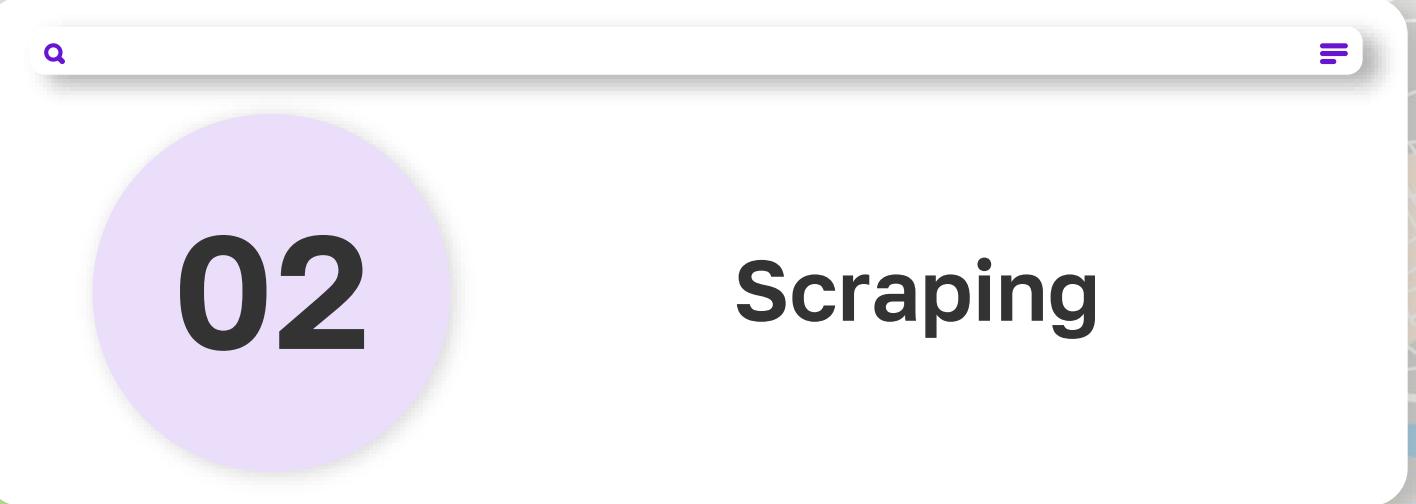


4

## Hypothesen

Es gibt eine signifikante Korrelation zwischen bestehendem Fluglärm und höheren Mietpreisen in diesen Gebieten.





**02**

# Scraping

```
import requests
from bs4 import BeautifulSoup
import re
import csv
import os
import time

# URL der Webseite, die wir scrauen wollen
base_url = 'https://www.immoscout24.ch/de/immobilien/mieten/kanton-zuerich'
page_num_suffix = '?pn='
page_num = 1

# Array, um die gesammelten Daten zu speichern
properties = []

while True:
    # URL für die aktuelle Seite erstellen
    url = base_url + (page_num_suffix + str(page_num) if page_num > 1 else '')

    print(f'Suche auf URL {url}')

    # Anfrage versenden
    response = requests.get(url)

    # Überprüfen, ob die Anfrage erfolgreich war
    if response.status_code != 200:
        print(f" Fehler beim Abrufen der Seite {url}. Statuscode: {response.status_code}")
        break

    # BeautifulSoup-Objekt erstellen, um die Seite zu parsen
    soup = BeautifulSoup(response.text, 'html.parser')

    # Immobiliencontainer auf der Seite finden
    property_list = soup.find_all('div', class_='HgListingCard_info_RKrwz')

    # Überprüfen, ob Immobilien gefunden wurden
    if not property_list:
        print('Keine Elemente gefunden, letzte Seite wahrscheinlich erreicht')
        break
```

- Import von erforderlichen Modulen wie requests, BeautifulSoup, und csv.
- Basis URL für Immobilienanzeigen im Kanton Zürich
- While-Schleife läuft, so lange Immobilienanzeigen vorhanden sind.

```
# Immobilien durchlaufen und Daten sammeln
for property in property_list:
    # Titel, Preis, Zimmer, etc. extrahieren.
    title = property.find('div', class_='HgListingRoomsLivingSpacePrice_roomsLivingS')

    # Adresse extrahieren
    address_raw = property.find('address').text.strip() if property.find('address')

    # Reguläre Ausdrücke für die gesuchten Daten innerhalb der Elemente
    zimmer_pattern = re.compile(r'(\d+\.\d*) Zimmer')
    quadratmeter_pattern = re.compile(r'(\d+)m²')
    preis_pattern = re.compile(r'CHF ([\d,]+)')
    address_pattern = re.compile(r'(.+), (\d{4}) (.+)')

    # Suche nach den Daten im String
    zimmer = zimmer_pattern.search(title)
    quadratmeter = quadratmeter_pattern.search(title)
    preis = preis_pattern.search(title)
    match = address_pattern.search(address_raw)

    # Extrahiere und konvertiere die Daten
    zimmer = float(zimmer.group(1)) if zimmer else None
    quadratmeter = int(quadratmeter.group(1)) if quadratmeter else None
    preis = int(preis.group(1).replace(',', '')) if preis else None
    address = match.group(1) if match else None
    plz = match.group(2) if match else None
    ort = match.group(3) if match else None
```

- Das Skript durchläuft eine Schleife, die die Seiten der Immobilienanzeigen abruft und die Daten extrahiert.
- Die Schleife findet die Immobilienanzeigen auf der Seite, extrahiert relevante Informationen wie Titel, Adresse, Zimmerzahl, Quadratmeter und Preis, und speichert sie in einem Dictionary.

```
# Extrahierten Daten zum Array hinzufügen
properties.append({
    'Zimmer': zimmer,
    'Quadratmeter': quadratmeter,
    'Preis': preis,
    'Adresse': address,
    'PLZ': plz,
    'Ort': ort
})

# Nächste Seite vorbereiten
page_num += 1
# Eine Pause einfügen, um zu vermeiden, die Website zu überlasten
time.sleep(1)

# Ausgabe der gesammelten Daten
for prop in properties:
    print(prop)

# Pfad zur Zieldatei
dir_name = os.path.dirname(__file__)
file_path = os.path.join(dir_name, 'wohnungen_raw.csv')

# Zieldatei befüllen
with open(file_path, mode='w', newline='', encoding='utf-8') as file:
    writer = csv.DictWriter(file, fieldnames=properties[0].keys())

    writer.writeheader()

    for prop in properties:
        writer.writerow(prop)

print("Scraping abgeschlossen. Daten wurden in die CSV-Datei geschrieben.")
```

- Die extrahierten Daten werden der Liste **properties** hinzugefügt.
- Das Skript macht eine Pause von 1 Sekunde zwischen den Anfragen, um eine Überlastung der Website zu vermeiden.
- Nach Abschluss des Scrapings werden die gesammelten Daten in eine CSV-Datei geschrieben.
- Die CSV-Datei wird im selben Verzeichnis wie das Skript erstellt.
- Insgesamt extrahiert und speichert der Code Immobiliendaten von ImmoScout24 für den Kanton Zürich in einer CSV-Datei, die dann für weitere Analysen oder Verarbeitungen verwendet werden können.



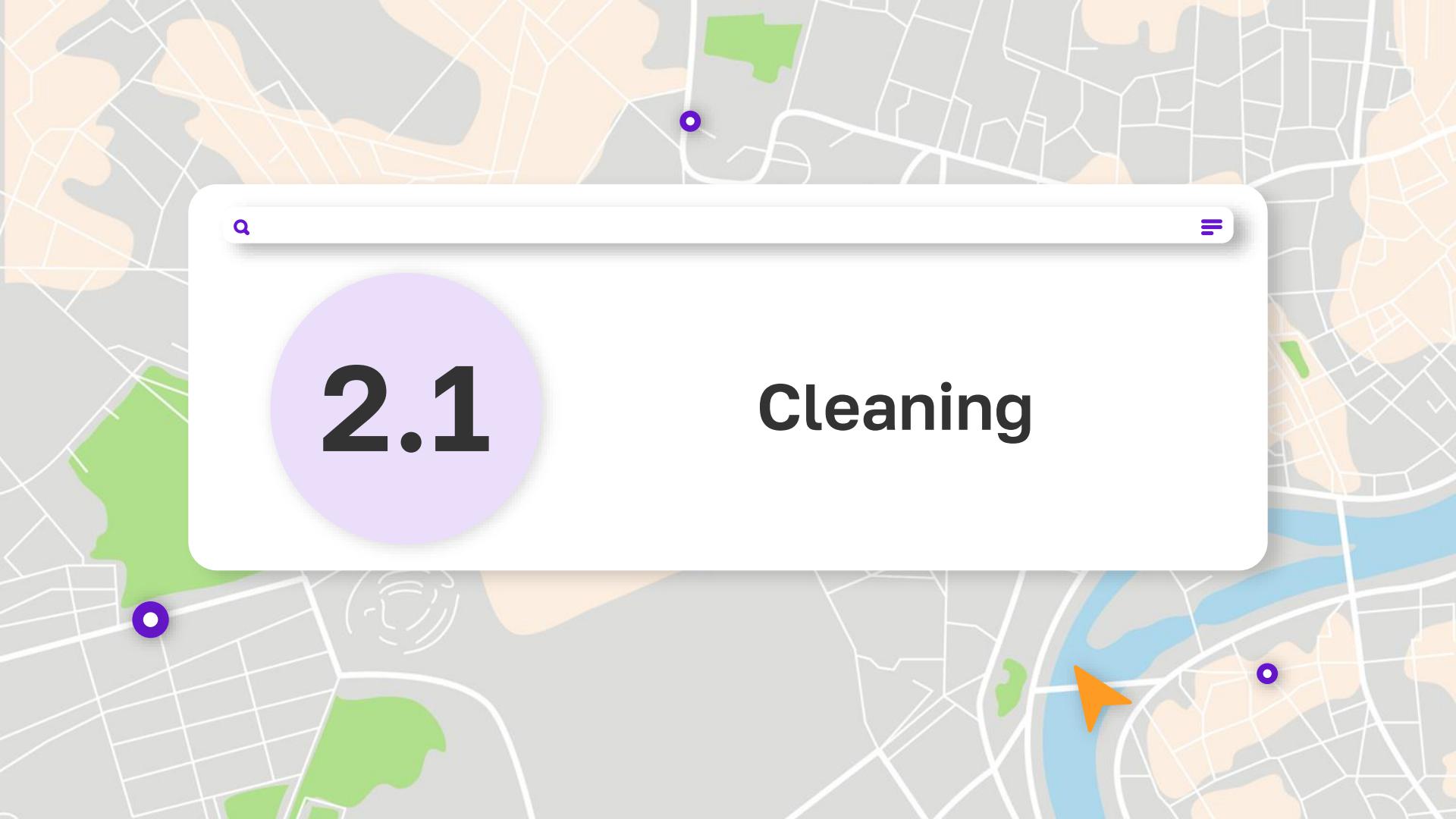
# CSV-Datei



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar labeled 'EXPLORER' and 'PROJECT'. Under 'PROJECT', there are several items: '.venv', '1\_scraping' (which contains 'cleaning.py' and 'scraping.py'), 'wohnungen\_clean....', 'wohnungen\_raw.csv' (which is currently selected), 'correlation and ml', and 'qgis\_output\_csv'. The main area displays the contents of 'wohnungen\_raw.csv'. The data consists of 24 rows, each representing a real estate listing with the following columns: Zimmer (Rooms), Quadratmeter (Square meters), Preis (Price), Adresse (Address), PLZ (Postcode), and Ort (City). The data is as follows:

	Zimmer	Quadratmeter	Preis	Adresse	PLZ	Ort
1	3.5,38,3560	"Haldenstrasse 169, 8003 Zürich",8055,Zürich				
2	2.5,54,4600	Haldenstrasse 167,8055,Zürich				
3	2.5,32,3190	"Haldenstrasse 169, 8003 Zürich",8055,Zürich				
4	3.5,180,8160	Sonnentalstrasse 13,8600,Dübendorf				
5	3.5,49,6050	Rosengartenstrasse 55,8037,Zürich				
6	3.5,150,3470	Trüllikerstrasse 9,8461,Oerlingen				
7	2.5,58,4550	"Allmannstrasse 55, 8052 Zürich",8052,Zürich				
8	2.5,,2740	Unterdorf 5,8453,Alten				
9	3.5,37,3650	Aemtlerstrasse 186,8003,Zürich				
10	2.5,54,3450	Breitistrasse 1,8335,Hittnau				
11	3.5,45,3320	Eggbühlstrasse 9,8050,Zürich				
12	2.5,57,3240	Eichhölzlistrasse 14,8192,Glattfelden				
13	3.5,49,4770	Rosengartenstrasse 55,8037,Zürich				
14	3.5,130,4400	Im Steinacher 3,8303,Bassersdorf				
15	3.5,160,5240	Im Steinacher 3,8303,Bassersdorf				
16	2.5,120,4750	Spindelstrasse 7,8041,Zürich				
17	3.5,63,2340	Waldeggweg 1,8302,Kloten				
18	2.5,62,6410	Ankerstrasse 114,8004,Zürich				
19	2.0,,1630	Badenerstrasse 54,8953,Dietikon				
20	3.5,140,4240	Waldeggweg 1,8302,Kloten				
21	3.5,290,6350	Niggitalstrasse,8630,Rüti				
22	2.0,29,4510	Zelgstrasse 19,8003,Zürich				
23	2.5,,3430,,					
24						

- Die gescrapte Daten werden in der Datei `wohnungen_raw.csv` gespeichert und können später für weitere Aufgaben verwendet werden.



**2.1**

# Cleaning

```
import pandas as pd
import os

def replace_umlaute(text):
    """ Ersetzt deutsche Umlaute in einem gegebenen Text. """
    replacements = {
        'ä': 'ae',
        'ü': 'ue',
        'ö': 'oe',
        'Ä': 'Ae',
        'Ü': 'Ue',
        'Ö': 'Oe'
    }
    for search, replace in replacements.items():
        text = text.replace(search, replace)
    return text

# Setze den absoluten Pfad zur CSV-Datei
dir_name = os.path.dirname(os.path.abspath(__file__))
file_path_raw = os.path.join(dir_name, 'wohnungen_raw.csv')
file_path_clean = os.path.join(dir_name, 'wohnungen_clean.csv')

# Versuche, die Daten einzulesen
try:
    data = pd.read_csv(file_path_raw)
    print("Daten erfolgreich geladen!")
except FileNotFoundError:
    print(f"Datei nicht gefunden: {file_path_raw}")
    exit()

# Überprüfe auf fehlende Werte und zeige sie an
print("Anzahl fehlender Werte pro Spalte:")
print(data.isnull().sum())

# Fehlende Werte entfernen: Betrachte nur die notwendigen Spalten
data = data.dropna(subset=['Quadratmeter', 'Preis', 'Adresse', 'PLZ', 'Ort'])

# Duplikate entfernen
data = data.drop_duplicates()
```

- Die Module 'pandas' (für die Datenverarbeitung) und 'os' (zur Handhabung von Dateipfaden) werden importiert.
- Die Funktion 'replace\_umlaute' ersetzt die deutschen Umlaute in einem gegebenen Text durch ASCII-Äquivalente.
- Der absolute Pfad zur Rohdaten-CSV-Datei und zur bereinigten CSV-Datei wird festgelegt.
- Das Skript versucht, die Rohdaten aus der CSV-Datei zu laden, zeigt eine Fehlermeldung an, falls die Datei nicht gefunden wird, und beendet sich dann.
- Fehlende Werte in den erforderlichen Spalten ('Quadratmeter', 'Preis', 'Adresse', 'PLZ', 'Ort') werden entfernt, und Duplikate werden entfernt.

```
# Konvertiere Datentypen, sicherstellen, dass keine Konvertierungsfehler auftreten
try:
    data['Quadratmeter'] = data['Quadratmeter'].astype(int)
    data['Preis'] = data['Preis'].astype(int)
    data['PLZ'] = data['PLZ'].astype(int)
except ValueError as e:
    print("Fehler bei der Datentypkonvertierung:", e)
    exit()

# Ersetze Umlaute in allen textbasierten Spalten
for column in data.select_dtypes(include=[object]).columns:
    data[column] = data[column].apply(replace_umlauts)

# Berechne den Preis pro Quadratmeter und runde auf zwei Dezimalstellen
data['Preis_pro_Quadratmeter'] = (data['Preis'] / data['Quadratmeter']).round(2)

# Kombiniere 'Adresse' und 'PLZ' in ein neues Attribut 'Adresse_PLZ'
data['Adresse_PLZ'] = data['Adresse'] + ', ' + data['PLZ'].astype(str)

# Füge das neue Attribut 'Kanton' hinzu und setze es auf 'Zurich' für alle Einträge
data['Kanton'] = 'Zurich'

# Definiere die gewünschte Spaltenreihenfolge
columns_order = ['Zimmer', 'Quadratmeter', 'Preis', 'Preis_pro_Quadratmeter', 'Adresse', 'PLZ', 'Ort', 'Kanton', 'Adresse_PLZ']

# Reorganisiere die Spalten gemäß der festgelegten Reihenfolge
data = data[columns_order]

# Überprüfe die bereinigten Daten und Datentypen
print("Bereinigte Daten:")
print(data.head())
print("\nDatentypen der Spalten:")
print(data.dtypes)

# Speichere die bereinigten Daten in einer neuen CSV-Datei
data.to_csv(file_path_clean, index=False)
print(f"Bereinigung abgeschlossen. Daten wurden in '{file_path_clean}' geschrieben.")
```

- Der Preis pro Quadratmeter wird berechnet und auf zwei Dezimalstellen gerundet.
- Eine neue Spalte 'Adresse\_PLZ' wird erstellt, indem 'Adresse' und 'PLZ' kombiniert werden.
- Eine neue Spalte 'Kanton' wird hinzugefügt und für alle Einträge auf 'Zurich' gesetzt.
- Die Reihenfolge der Spalten wird neu organisiert.
- Die bereinigten Daten und ihre Datentypen werden angezeigt.
- Die bereinigten Daten werden in einer neuen CSV-Datei gespeichert.



# CSV-Datei



wohnungen\_clean.csv ×

1\_scraping > wohnungen\_clean.csv

```
1 Zimmer,Quadratmeter,Preis,Preis_pro_Quadratmeter,Adresse,PLZ,Ort,Kanton,Adresse_PLZ
2 3.5,38,3560,93.68,"Haldenstrasse 169, 8003 Zuerich",8055,Zuerich,Zurich,"Haldenstrasse 169, 8003 Zuerich, 8055"
3 2.5,54,4600,85.19,Haldenstrasse 167,8055,Zuerich,Zurich,"Haldenstrasse 167, 8055"
4 2.5,32,3190,99.69,"Haldenstrasse 169, 8003 Zuerich",8055,Zuerich,Zurich,"Haldenstrasse 169, 8003 Zuerich, 8055"
5 3.5,180,8160,45.33,Sonnentalstrasse 13,8600,Duebendorf,Zurich,"Sonnentalstrasse 13, 8600"
6 3.5,49,6050,123.47,Rosengartenstrasse 55,8037,Zuerich,Zurich,"Rosengartenstrasse 55, 8037"
7 3.5,150,3470,23.13,Truellikerstrasse 9,8461,Oerlingen,Zurich,"Truellikerstrasse 9, 8461"
8 2.5,58,4550,78.45,"Allmannstrasse 55, 8052 Zuerich",8052,Zuerich,Zurich,"Allmannstrasse 55, 8052 Zuerich, 8052"
9 3.5,37,3650,98.65,Aemtlerstrasse 186,8003,Zuerich,Zurich,"Aemtlerstrasse 186, 8003"
10 2.5,54,3450,63.89,Breitistrasse 1,8335,Hittnau,Zurich,"Breitistrasse 1, 8335"
11 3.5,45,3320,73.78,Eggbuehlstrasse 9,8050,Zuerich,Zurich,"Eggbuehlstrasse 9, 8050"
12 2.5,57,3240,56.84,Eichhoelzlistrasse 14,8192,Glattfelden,Zurich,"Eichhoelzlistrasse 14, 8192"
13 3.5,49,4770,97.35,Rosengartenstrasse 55,8037,Zuerich,Zurich,"Rosengartenstrasse 55, 8037"
14 3.5,130,4400,33.85,Im Steinacher 3,8303,Bassersdorf,Zurich,"Im Steinacher 3, 8303"
15 3.5,160,5240,32.75,Im Steinacher 3,8303,Bassersdorf,Zurich,"Im Steinacher 3, 8303"
16 2.5,120,4750,39.58,Spindelstrasse 7,8041,Zuerich,Zurich,"Spindelstrasse 7, 8041"
17 3.5,63,2340,37.14,Waldeggweg 1,8302,Kloten,Zurich,"Waldeggweg 1, 8302"
18 2.5,62,6410,103.39,Ankerstrasse 114,8004,Zuerich,Zurich,"Ankerstrasse 114, 8004"
```



**03**

**QGIS**





# MMQGIS Plugin

## mmqgis

### A collection of QGIS vector layer operation plugins

MMQGIS is a set of Python plugins for manipulating vector map layers in Quantum GIS: CSV input/output/join, geocoding, geometry conversion, buffering, hub analysis, simplification, column modification, and simple animation. MMQGIS provides an alternative to the Processing toolbox, with verbose progress reporting, an intuitive user interface, direct shapefile/CSV-file access, and some additional capabilities missing from other plugin sets.

★★★★★ 421 rating vote(s), 1342244 downloads

**Category** Vector

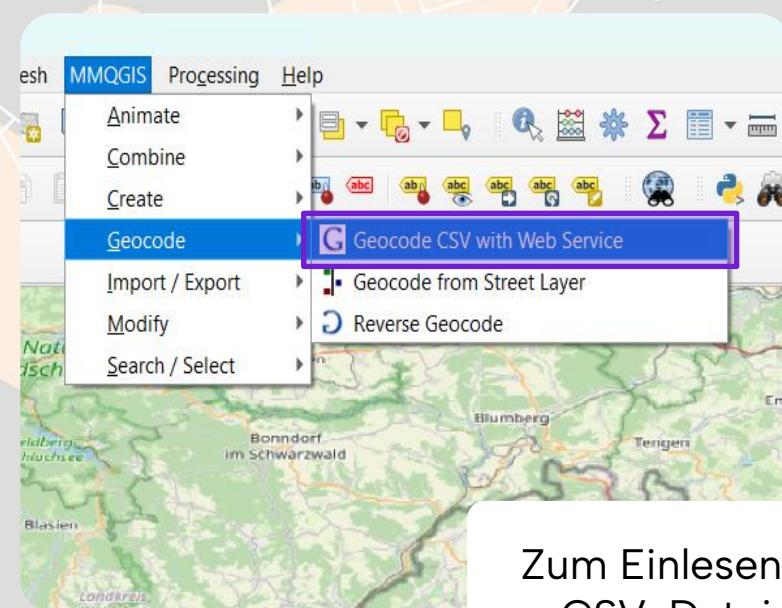
**Tags** vector, sort, merge, animate, delete

**More info** [homepage](#) [bug tracker](#) [code repository](#)

**Author** Michael Minn

**Installed version** 2021.9.10

**Available version (stable)** 2021.9.10 updated at Fri Sep 10 20:12:36 2021 GMT



Zum Einlesen der  
CSV-Dateien



# Wohnungen importieren



Web Service Geocode

Input CSV File (UTF-8)  
C:\Users\ranuj\Downloads\wohnungen\_clean.csv

Address City  
Adresse\_PLZ Ort

State Country

Canton Country

Web Service  
OpenStreetMap / Nominatim

API Key

Duplicate Handling  
Use Only First Result

Output File Name  
er/Einsatz von Geodaten im Marketing/Abschlussprojekt/Fluglaerm/temp3.shp

Not Found Output List  
er/Einsatz von Geodaten im Marketing/Abschlussprojekt/Fluglaerm/temp3.csv

20 of 868 = 17 found

Close Apply

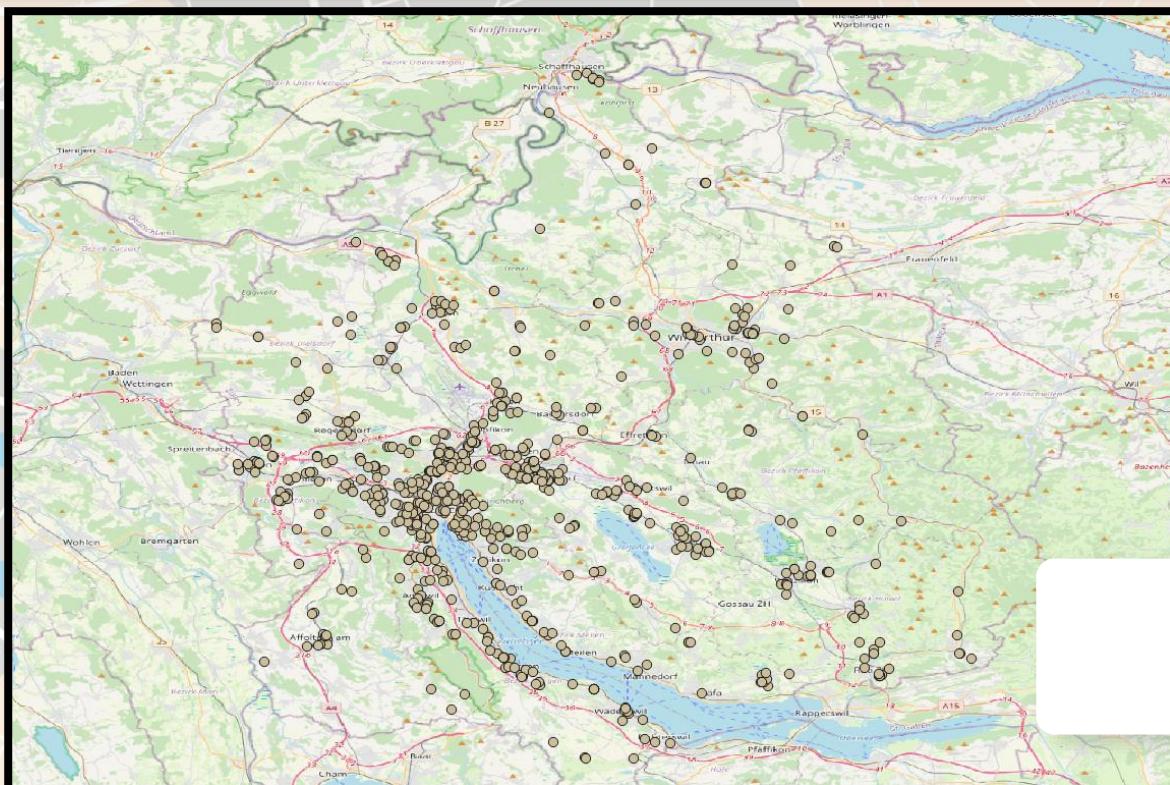
Zimmer	Quadratmeter	Preis
3.5	38	
2.5	54	
2.5	32	
3.5	180	
3.5	49	
3.5	150	
2.5	58	
3.5	37	
2.5	54	
3.5	45	
2.5	57	
2.5	40	

## Ursachen für nicht gefundene Wohnungen:

- Umlaute Ä,Ö,Ü
- Abkürzungen wie Str.
- Sonderzeichen
- Fehlende Informationen (z.B Hausnummer)



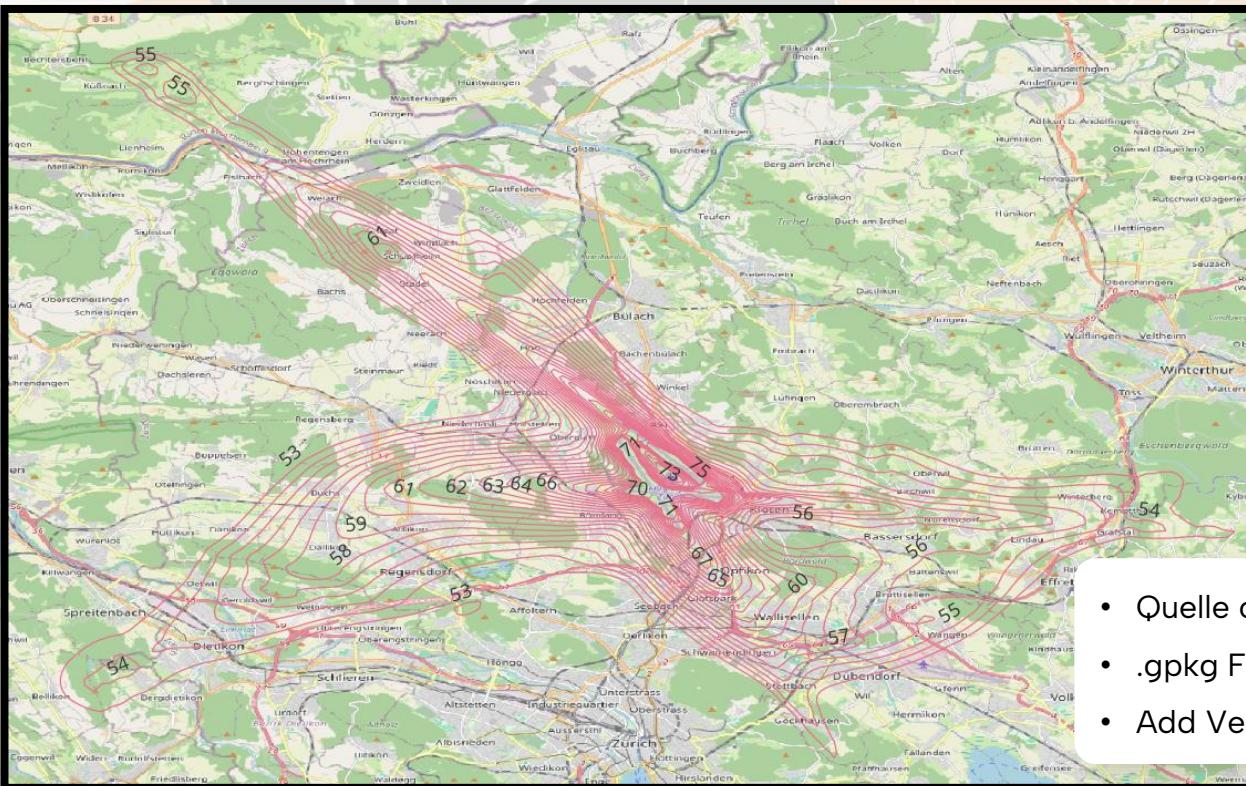
# Wohnungen importieren



Ergebnis



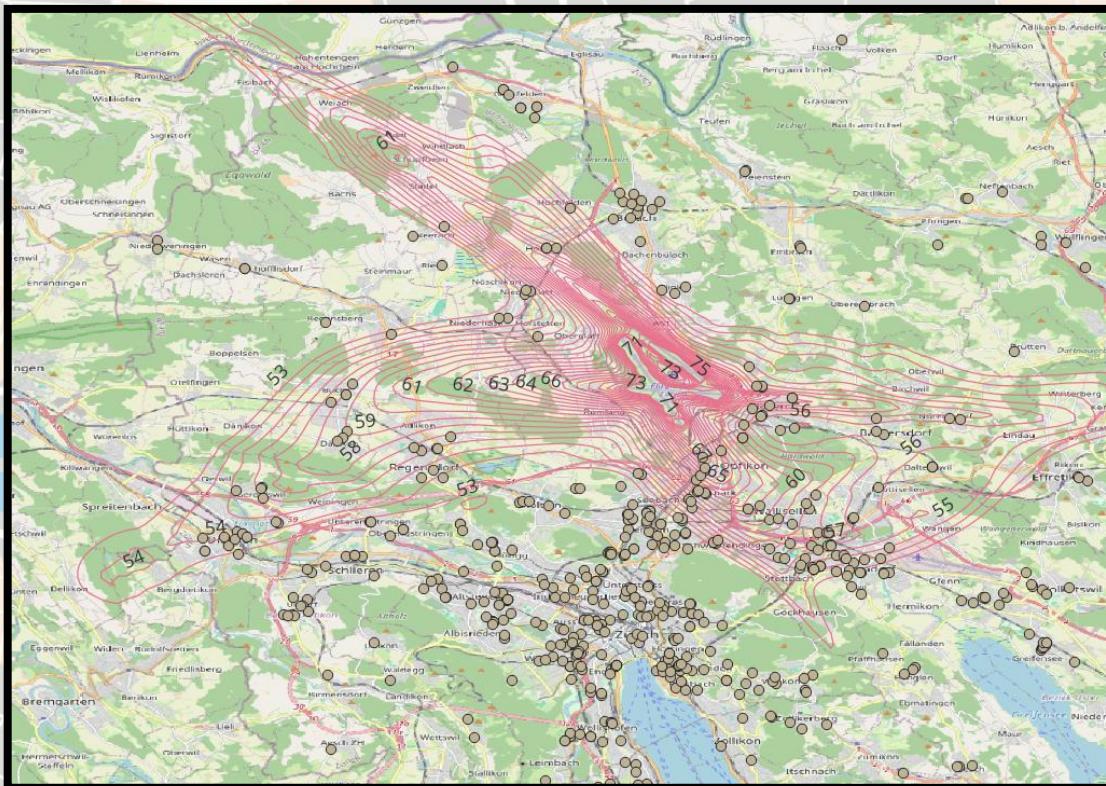
# Fluglärm Zonen



- Quelle opendata.swiss
  - .gpkg File
  - Add Vector Layer



# Join Attributes



## Processing toolbox

Hier werden die beiden Layer gejoint



# Attributen mergen



Join Attributes by Nearest

Parameters Log

**Input layer**  
Wohnung\_mit\_Larm [EPSG:4326]  
 Selected features only

**Input layer 2**  
laermbelastungskataster-zivilflugplaetze\_2056 — OverallTrafficDay\_Lr [EPSG:2056]  
 Selected features only

**Layer 2 fields to copy (leave empty to copy all fields) [optional]**  
0 field(s) selected  
 Discard records which could not be joined

**Joined field prefix [optional]**  
[empty input field]

**Maximum nearest neighbors**  
1

**Maximum distance [optional]**  
Not set degrees

**Joined layer [optional]**  
[Create temporary layer]

0%

Advanced Run as Batch Process... Run Cancel Close Help

**Join attributes by nearest**

This algorithm takes an input vector layer and creates a new vector layer that is an extended version of the input one, with additional attributes in its attribute table.

The additional attributes and their values are taken from a second vector layer, where features are joined by finding the closest features from each layer. By default only the single nearest feature is joined, but optionally the join can use the n-nearest neighboring features instead. If multiple features are found with identical distances these will all be returned (even if the total number of features exceeds the specified maximum feature count).

If a maximum distance is specified, then only features which are closer than this distance will be matched.

The output features will contain the selected attributes from the nearest feature, along with new attributes for the distance to the near feature, the index of the feature, and the coordinates of the closest point on the input feature (`feature_x`, `feature_y`) to the matched nearest feature, and the coordinates of the closest point on the

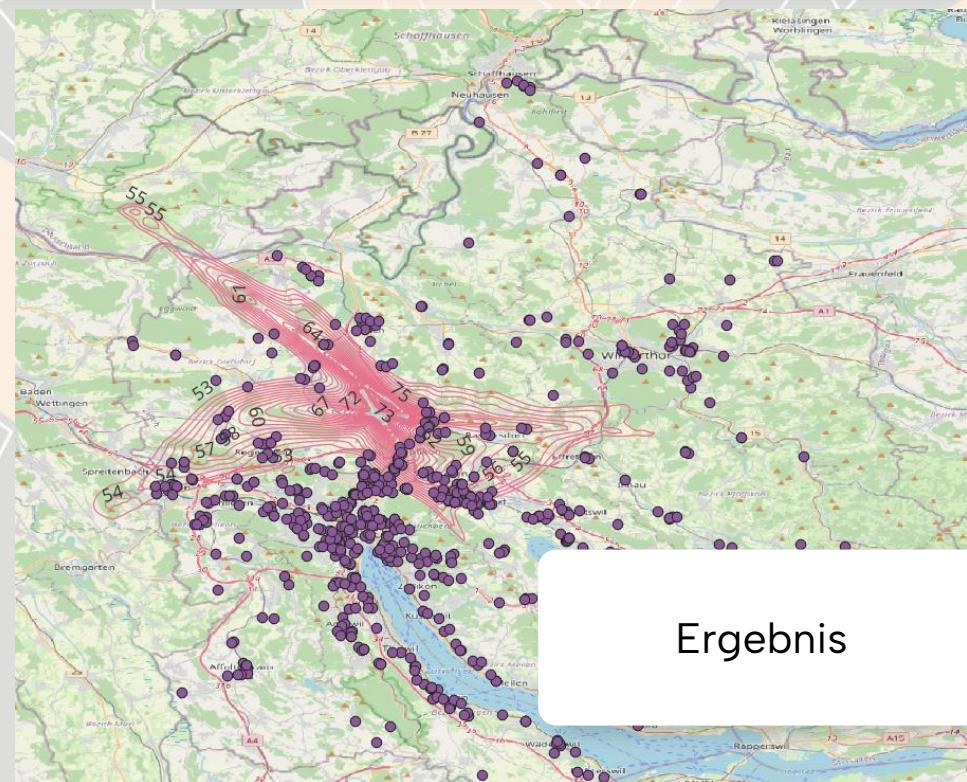


# Joined Layer

Layers



- Wohnungen\_mit\_Laerm
- Wohnung\_mit\_Larm
- Fluglaerm\_ganzerKanton
- Joined layer
- Joined layer
- temp2
- laermbelastungskataster-zivilflugplaetze
- OpenStreetMap





# Joined Layer



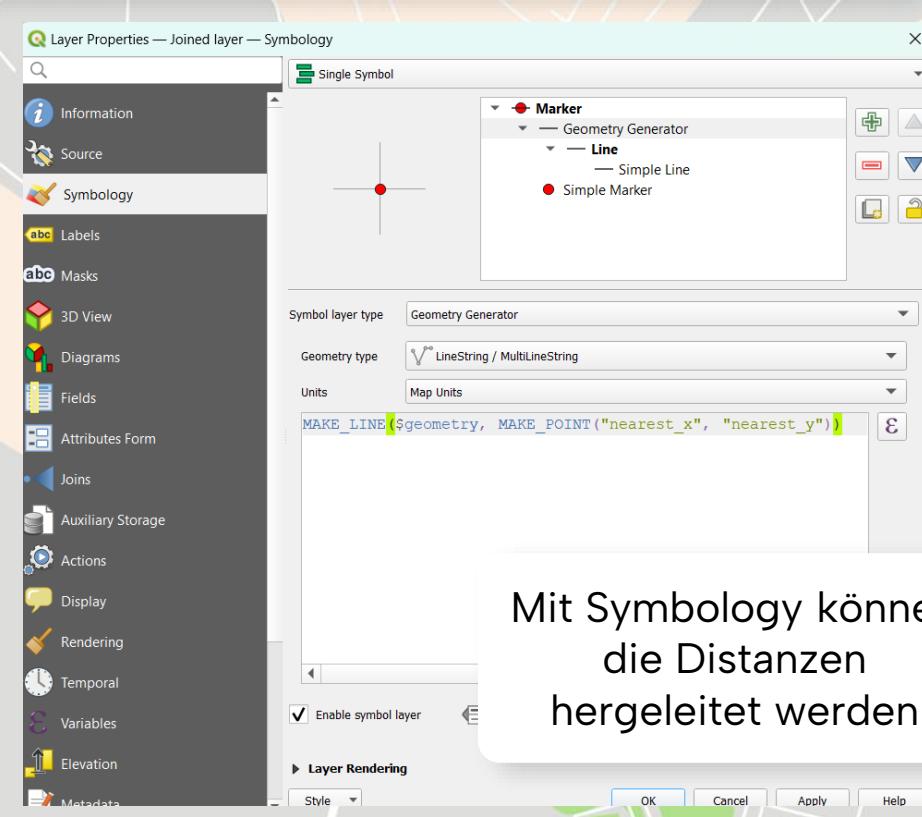
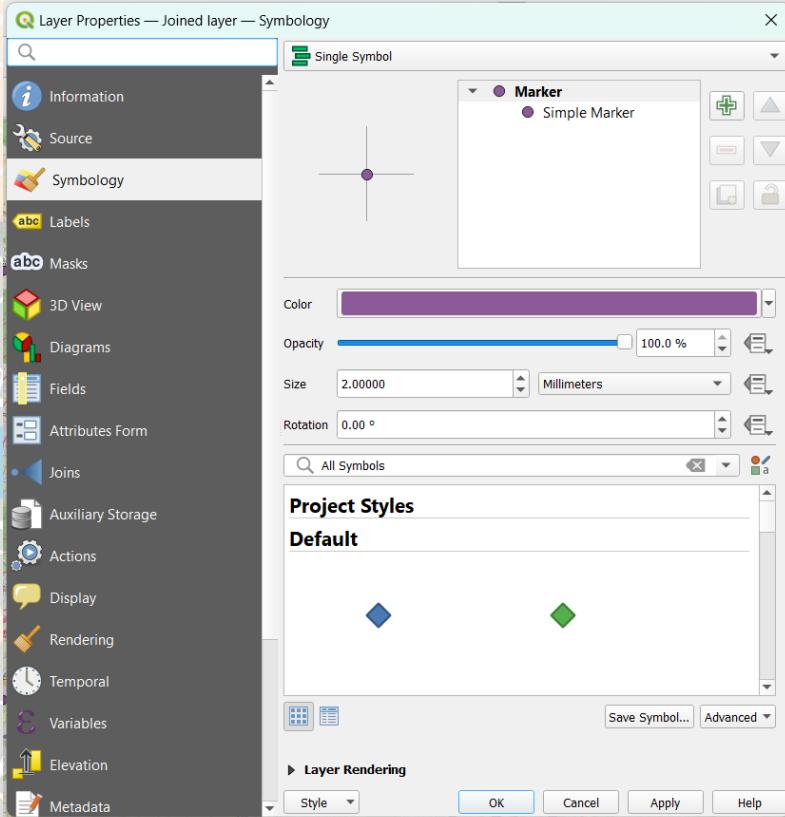
Joined layer — Features Total: 818, Filtered: 818, Selected: 0

option_en	typ_sortseq	typ_active	pub_oid	pub_publicationdate	typ_pub_oid	legalstatus_oid	n	distance	feature_x	feature_y	nearest_x	nearest_y
1	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	05380407953818594	8.511324695437034	47.37028035	8.48599946614058	47.41775151741806
2	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	0.001404056569566...	8.6019996	47.3968744	8.60261656301019	47.39561315795559
3	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	02960749568714156	8.52916275991669	47.39586895	8.547824268823687	47.41885485625097
4	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	15826596390447564	8.6757541	47.6252418	8.630343159225243	47.47363058079586
5	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	0.046000603924715...	8.508438995854974	47.37758770000001	8.486687183995505	47.418120578534214
6	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	0.1190950695760818	8.823328845172277	47.35863265	8.740716354884945	47.44441616849777
7	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	0.004261111925311...	8.54443281	47.4161835	8.546830599626565	47.419632350599706
8	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	01754428115244072	8.5065999	47.5554852	8.49665826177846	47.54102955644366
9	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	02960749568714156	8.52916275991669	47.39586895	8.547824268823687	47.41885485625097
10	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	0.006421170993304...	8.6426479	47.4318969	8.640630717631124	47.437992998114034
11	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	0.006421170993304...	8.6426479	47.4318969	8.640630717631124	47.437992998114034
12	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	08150823353362029	8.5181717	47.3394016	8.5649448930159	47.40615411541337
13	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	0.000119117735162...	8.577384	47.4419715	8.57747254396543	47.4420511680618826
14	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	0.04588803759981...	8.5280816	47.3762772	8.555457884361221	47.413105468559564
15	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	0.000119117735162...	8.577384	47.4419715	8.57747254396543	47.4420511680618826
16	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	22383678131848286	8.8656731	4		
17	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	0.056497801342245...	8.521891775314465	47.3676597		
18	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	0.048002071870660...	8.55448780000011	47.3559191		
19	2	t	ch20bn9n00065030	20211201	ch20bn9n00065030	ch20bn9n00000002	1	04227392452537727	8.51617	4		
41												

Geodaten als CSV



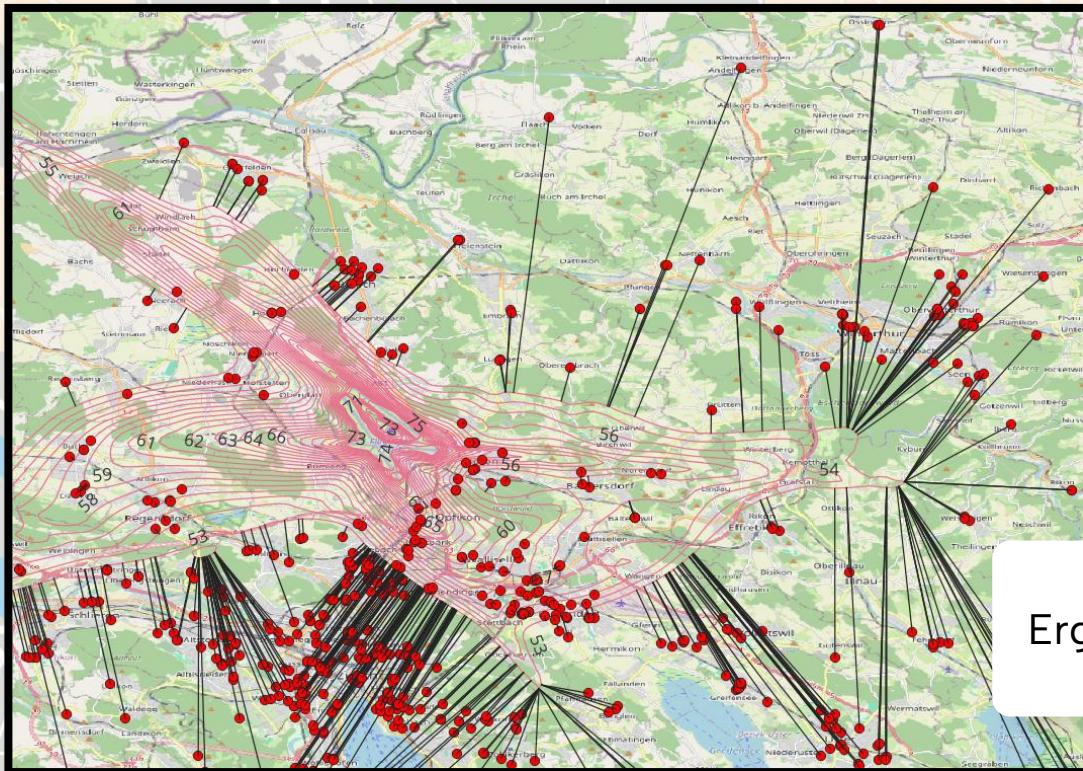
# Distanzen herleiten



Mit Symbology können  
die Distanzen  
hergeleitet werden



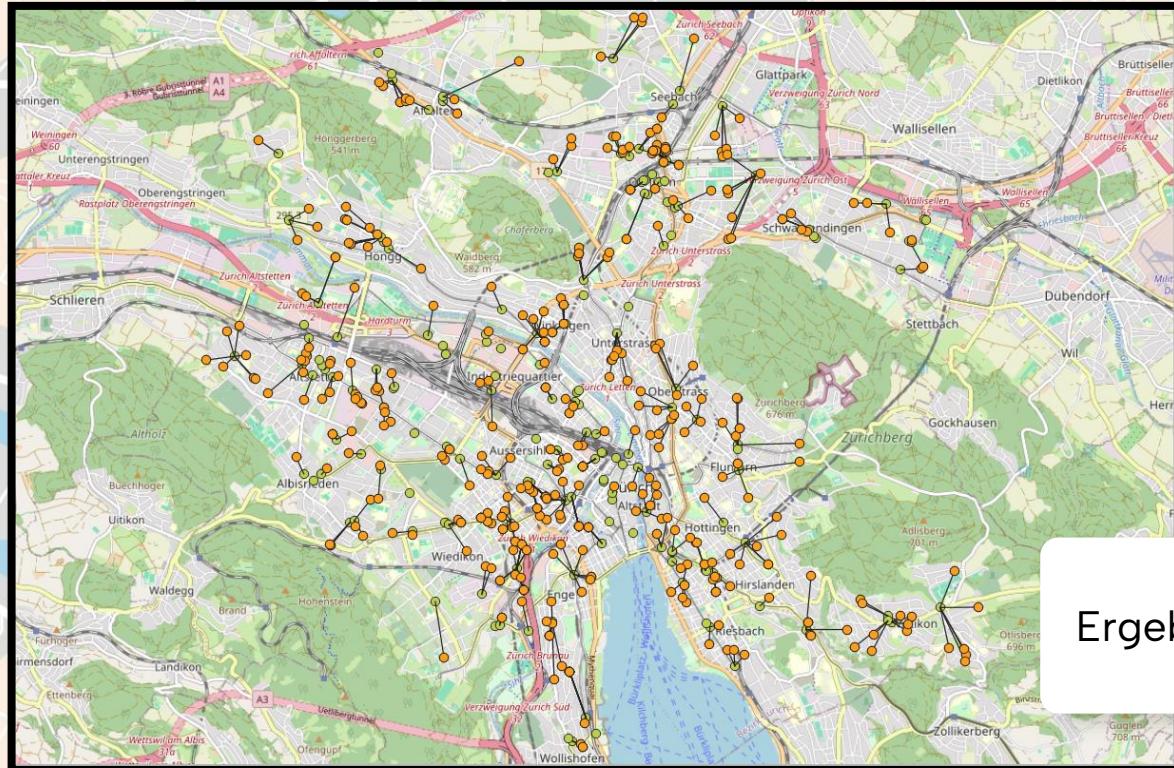
# Mietpreise <-> Fluglärm



Ergebnis (grafisch)



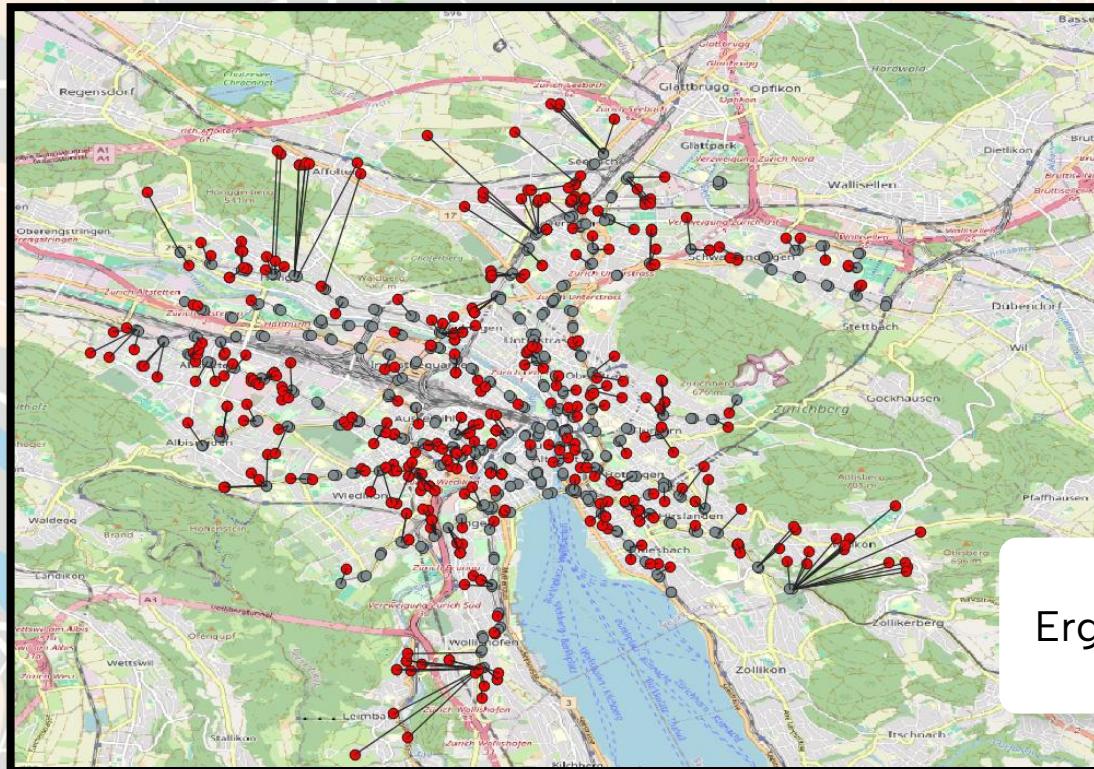
# Mietpreise <-> Supermärkte



Ergebnis (grafisch)



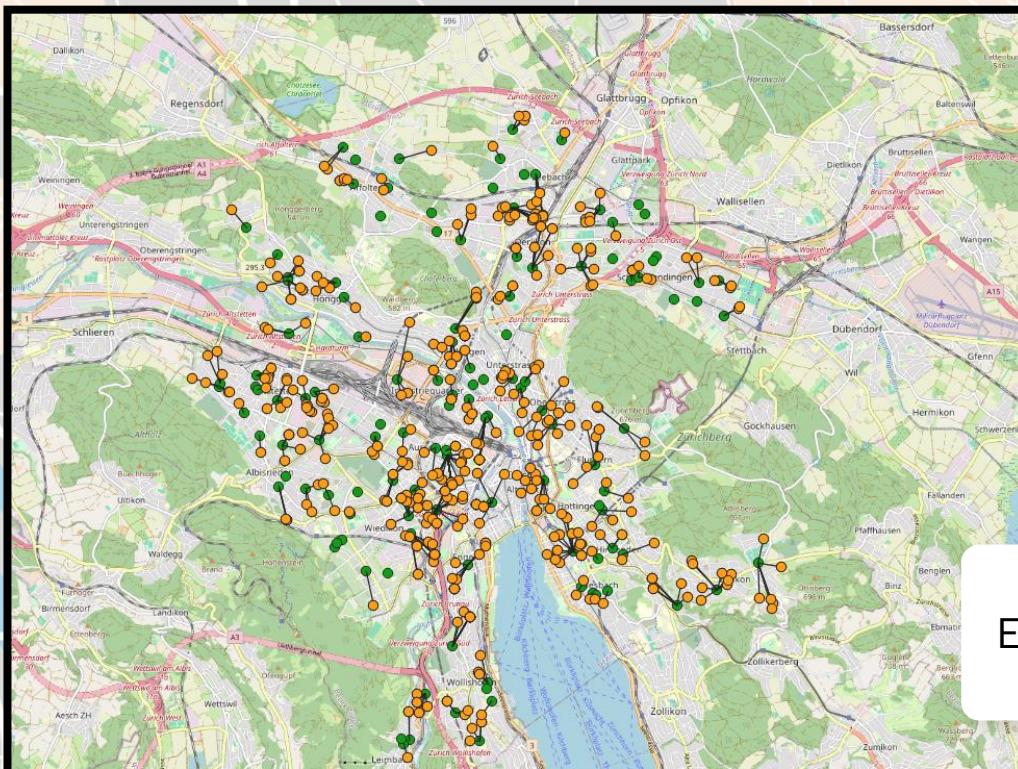
# Mietpreise <-> Tramsationen



Ergebnis (grafisch)



# Mietpreise <-> Volksschulen



Ergebnis (grafisch)



# 04

## Auswertung und Fazit



# Imports

```
import pandas as pd # Zum Laden und Manipulieren von Daten.  
import matplotlib.pyplot as plt # Zum Erstellen von Visualisierungen, insbesondere für die Korrelationsmatrix.  
import seaborn as sns # Zum Erstellen einer Heatmap.  
from sklearn.model_selection import train_test_split # Zum Aufteilen der Daten in Trainings- und Testsets.  
from sklearn.linear_model import LinearRegression # Zum Erstellen eines linearen Regressionsmodells.  
from sklearn.metrics import mean_squared_error, r2_score # Zum Bewerten des linearen Regressionsmodells.
```

- pandas: Zum Laden und Manipulieren von Daten.
- matplotlib.pyplot: Zum Erstellen von Visualisierungen.
- seaborn: Zum Erstellen von Heatmaps
- sklearn.model\_selection: Zum Aufteilen der Daten in Trainings- und Testsets.
- sklearn.linear\_model: Zum Erstellen eines linearen Regressionsmodells.
- sklearn.metrics: Zum Bewerten des linearen Regressionsmodells.



# Daten laden und cleanen



```
def load_data(filepath): # Lädt die Daten aus einer CSV-Datei in ein DataFrame.  
    return pd.read_csv(filepath)  
  
def preprocess_data(data): # Bereitet die Daten vor: entfernt Spalten ohne Daten und ersetzt fehlende Werte in numerischen Spalten.  
    data_clean = data.dropna(axis=1, how='all')  
    numerical_columns = data_clean.select_dtypes(include=['float64', 'int64']).columns  
    data_clean[numerical_columns] = data_clean[numerical_columns].fillna(data_clean[numerical_columns].median()) # Füllt fehlende Werte mit dem Median  
    return data_clean
```

- `load_data`: Lädt Daten aus einer CSV-Datei in ein pandas DataFrame.
- `preprocess_data`: Bereitet die Daten vor, indem Spalten ohne Daten entfernt und fehlende Werte in numerischen Spalten durch den Median ersetzt werden.



# Corr Matrix und LR

```
def plot_correlation_matrix(data, features): # Erstellt eine Korrelationsmatrix für die angegebenen Merkmale.
    correlation_matrix = data[features].corr()
    plt.figure(figsize=(10, 8))
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
    plt.title('Correlation Matrix of Features')
    plt.show()

def fit_linear_regression(X, y): # Passt ein lineares Regressionsmodell an die Daten an.
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    return model, mse, r2
```

- `plot_correlation_matrix`: Erstellt und visualisiert eine Korrelationsmatrix für die angegebenen Merkmale.
- `fit_linear_regression`: Passt ein lineares Regressionsmodell an die Daten an, teilt die Daten in Trainings- und Testsets, und bewertet die Modellleistung.



# Main-Funktion

```
def main(): # Main-Funktion, die die anderen Funktionen steuert.
    data = load_data('./agis_output_csv/Wohnungen_mit_Laerm.csv') # Lädt Daten.
    data_clean = preprocess_data(data) # Bereitet Daten vor.

    plot_correlation_matrix(data_clean, ['Zimmer', 'Quadratmet', 'Preis', 'LEVEL_DB']) # Zeigt Korrelationsmatrix.

    X = data_clean[['Zimmer', 'Quadratmet', 'LEVEL_DB']] # Merkmale für das Modell.
    y = data_clean['Preis'] # Zielvariable für das Modell.
    model, mse, r2 = fit_linear_regression(X, y) # Führt lineare Regression durch und erhält Modellmetriken.

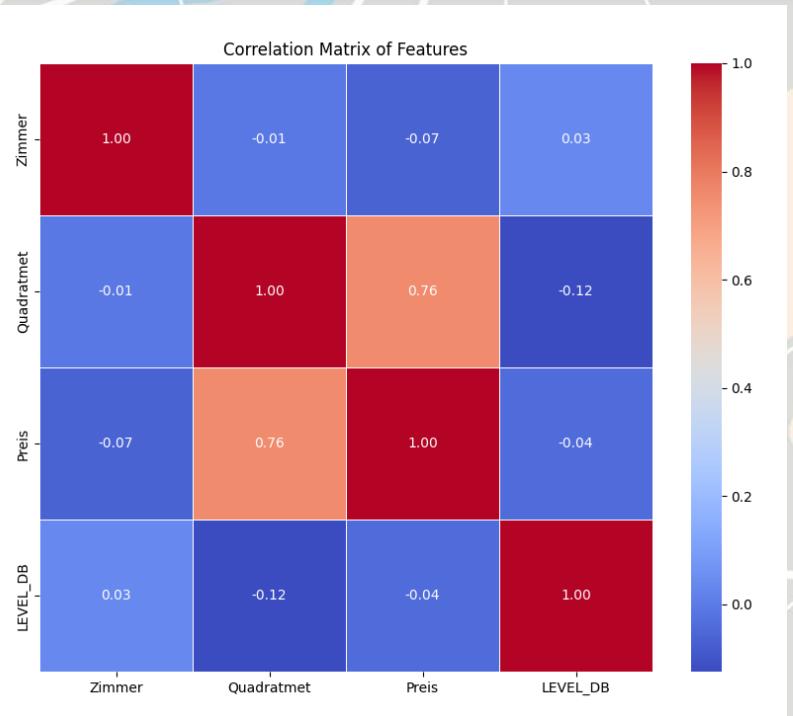
    print(f"Mean Squared Error: {mse}") # Gibt den MSE aus.
    print(f"R^2 Score: {r2}") # Gibt R^2-Wert aus.

if __name__ == "__main__":
    main() # Führt die Hauptfunktion aus, wenn das Skript direkt aufgerufen wird.
```

- Datenvorbereitung: Laden und Bereinigen der Daten aus einer CSV-Datei.
- Datenanalyse: Visualisierung der Korrelationen zwischen den Merkmalen.
- Modellierung: Training und Bewertung eines linearen Regressionsmodells zur Vorhersage des Preises basierend auf anderen Merkmalen.



# Auswertung Fluglärm



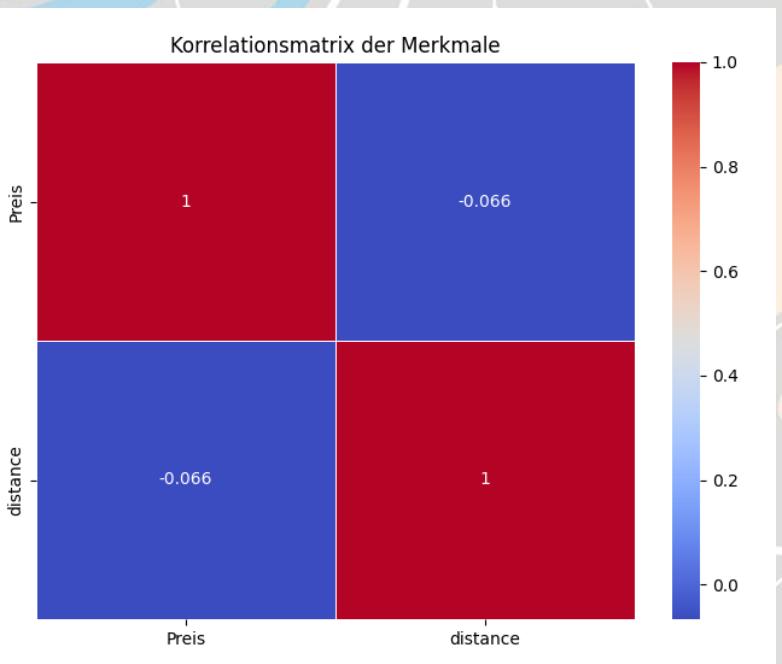
(Mietpreise Gemeinden rund um den Flughafen)

MSE: 1209846  
R2: 0.46

Erkenntnis: Sehr schwache negative Korrelation



# Supermärkte



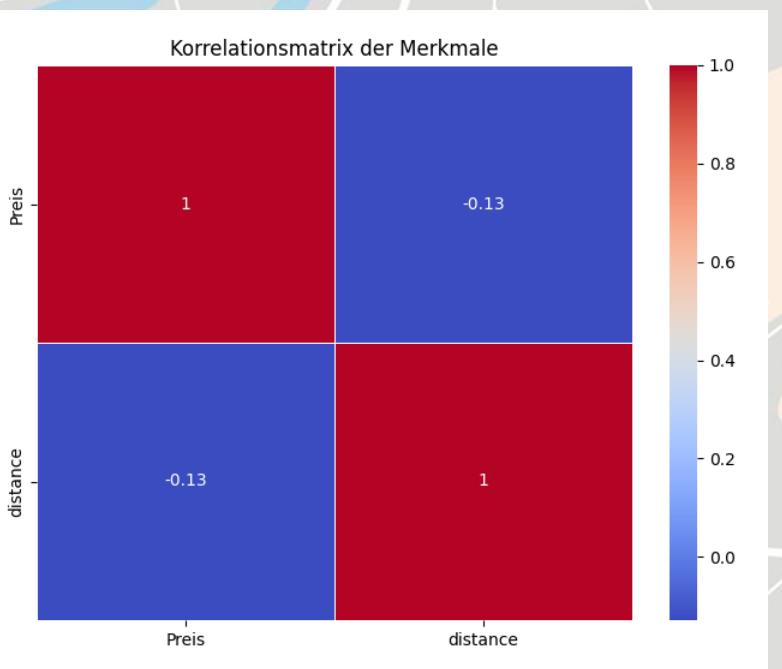
(Mietpreise Stadt Zürich)

MSE: 2861494  
R2: -0.03

Erkenntnis: Sehr schwache negative Korrelation



# Tramstationen



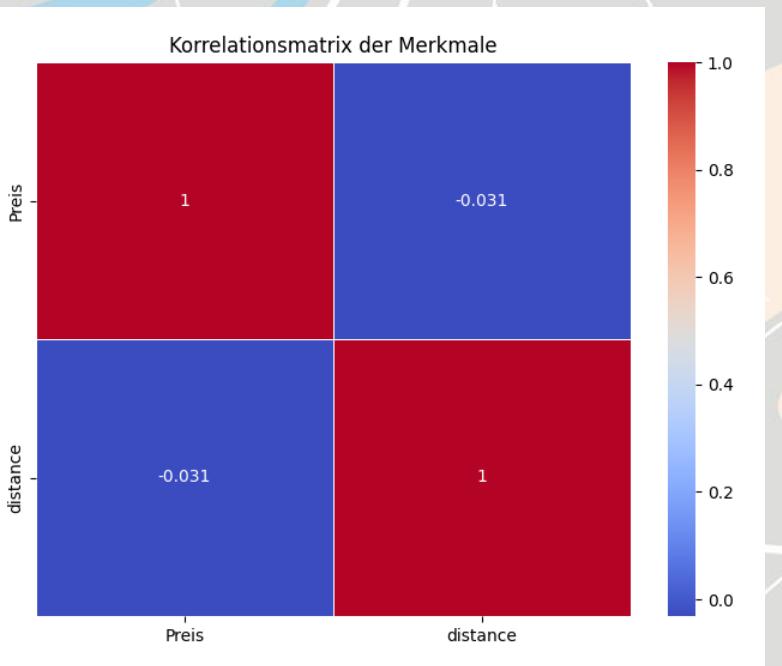
(Mietpreise Stadt Zürich)

MSE: 2847802  
R2: -0.02

Erkenntnis: Sehr schwache negative Korrelation



# Volksschulen



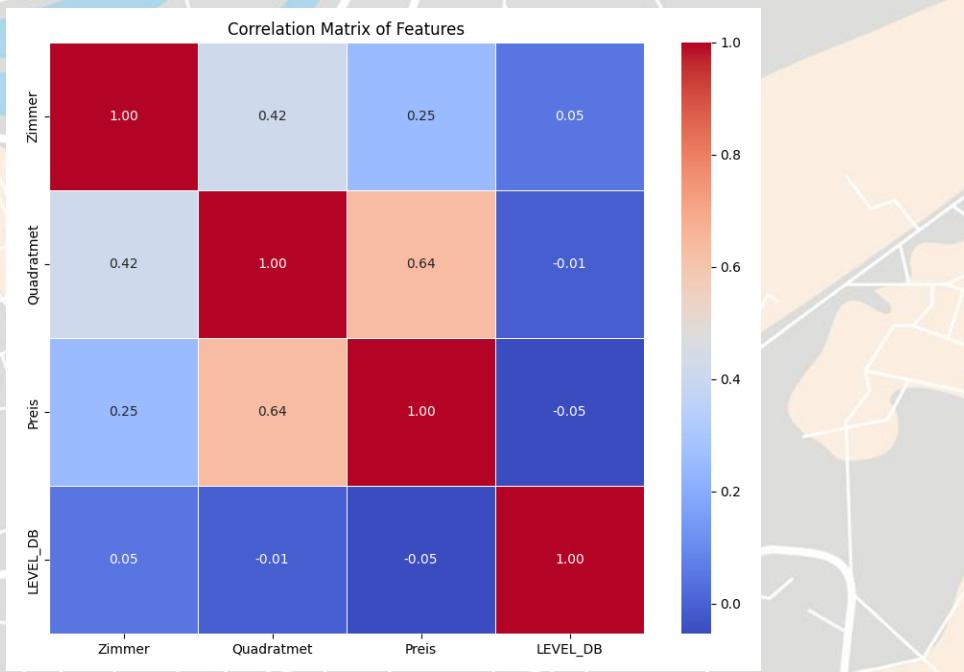
(Mietpreise Stadt Zürich)

MSE: 2887434  
R2: -0.04

Erkenntnis: Sehr schwache negative Korrelation



# Fluglärm Kanton ZH



(Mietpreise Ganzer Kanton Zürich)

MSE: 1519281  
R2: 0.41

Erkenntnis: Sehr schwache negative Korrelation



# Fazit



Die Entfernung zu Volksschulen, Supermärkten und Tramstationen sowie der Ausprägung des Fluglärms haben **kaum einen Einfluss** auf den Mietpreis.

Volksschulen, Supermärkte und Tramstationen:

In der Stadt gibt es eine **sehr dichte Infrastruktur**, wodurch die Nähe zu diesen Einrichtungen keinen signifikanten Einfluss auf die Mietpreise hat.

Fluglärm:

Der Fluglärm wird **nicht als grosses Problem** wahrgenommen und beeinflusst daher den Mietpreis nicht wesentlich.



# Schlusswort



Folgende Limitationen sind uns aufgefallen:  
Es liegen nicht die Mietpreise aller Wohnungen vor.  
Es fehlen Angaben zum Zustand der Wohnungen, was sicherlich von Bedeutung wäre.

Handlungsempfehlungen:  
Mehr Attribute einbeziehen, wie beispielsweise Baujahr und Schalldichte, da diese ebenfalls einen Einfluss haben auf den Mietpreis, und wie der Fluglärm wahrgenommen wird innerhalb der Wohnung.

Ausblick auf zukünftige Forschung:  
Vorhersage, in welchen Gebieten die Wohnungspreise steigen werden.



# Github Repo



Link:

<https://github.com/Denis-Machacka/EGM-Projekt-Gruppe2.git>