data mining project

May 31, 2023

1 Data Mining

2 SmartRentCalculator : Creating a Machine Learning-Based Tool for Predicting Rental Prices and Identifying Affordable Housing Options

3 TAPPWA PHILL MHEMBERE

3.1 MATRIC NUMBER:30006644

30-May-2023

3.2 1.Executive Summary

In this SmartRentCalculator project, I aimed to tackle the challenges that international students like myself face when searching for affordable and suitable off-campus housing while planning to study at Constructor University in Bremen, My primary focus was on creating a comprehensive and accurate dataset through web scraping techniques from the popular German rental website, eBay Kleinanzeigen.The main goal of my project was to develop a web-based tool Which i called SmartRentCalculator. I implemented machine learning algorithms to predict rental prices based on various features of a rental property, including location, number of bedrooms, square footage, and more. Additionally, I employed clustering and machine learning techniques to identify clusters of affordable rental properties based on the user's desired location and preferences.

The target users of the SmartRentCalculator are international students, including myself, who may not be familiar with the local rental market in Germany and require off-campus accommodations. My idea is that i designed the tool to be accessible through the Constructor University website, making it convenient for students to find affordable housing options. However, I also recognized that local students could benefit from the tool when seeking affordable housing alternatives.To achieve the objectives of my project, I had to employ web scraping techniques to extract data from eBay Kleinanzeigen. This process allowed me to compile a comprehensive and accurate dataset that would serve as the foundation for the SmartRentCalculator. I meticulously collected data on rental properties in Bremen, ensuring the dataset's quality and representativeness.Once I obtained the dataset, I embarked on a full cycle of data mining processes. I had to clean the data, addressing issues such as missing values, outliers, and inconsistencies. Data integration techniques were then applied to consolidate information from multiple sources, ensuring the dataset's compatibility and consistency.Next, I focused on data transformation, leveraging tools such as Python, Scikit-learn, and Pandas to implement and apply various data mining methods. Using advanced supervised ma-

chine learning techniques, I built models capable of predicting rental prices accurately. Evaluating and comparing different algorithms allowed me to select the most suitable ones based on factors like scalability, efficiency, and performance in practical scenarios.

Throughout the project, I continually refined the SmartRentCalculator to optimize its usability and effectiveness. In addition to predicting rental prices, I incorporated natural language processing (NLP) techniques to enhance the tool's functionality. By allowing users to describe specific details of their desired rental property, beyond standard features like bedrooms and square footage, I aimed to provide a more personalized and accurate housing recommendation system.The SmartRentCalculator project yielded significant outcomes. I successfully created a machine learning-based tool that predicts rental prices and identifies affordable housing options. Furthermore, the dataset I compiled comprised over 10,000 entries, ensuring its robustness and reliability. By incorporating NLP techniques, I provided users with a more comprehensive and precise housing search experience.And it is important to mention that this model is too good to be true because it's accuracy rate is almost hundred percent which raises more questions.Lastly let me say through this project, I have gained practical experience in performing a full cycle of data mining and analysis, and have hope to contributed to the well-being of the international student community at Constructor University by providing an accessible and user-friendly housing solution.

3.3 2.Introduction

Just trying to Wrap your head around, knowing how much of campus housing my Might cost or Finding affordable and suitable off-campus housing can be a daunting task for international students planning to study at Constructor University in Bremen.I understand the challenges and uncertainties that come with searching for housing in an unfamiliar city and navigating a foreign rental market. To address this issue, As i mentioned i undertook in the SmartRentCalculator project, with the primary focus on creating a comprehensive and accurate dataset through web scraping.The creation of a comprehensive and accurate dataset is of utmost importance in this project. Building a robust dataset lays the foundation for accurate predictions and reliable recommendations, essential for assisting international students in finding affordable and suitable off-campus housing in Bremen. By collecting and curating rental data from eBay Kleinanzeigen, the dataset captures crucial information about rental properties, including location, bedrooms, square footage, and amenities. This dataset serves as the key resource for training machine learning models to predict rental prices and identify clusters of affordable housing options. It enables the SmartRentCalculator to provide valuable insights and empower students to make informed decisions. Moreover, the dataset's value extends beyond this project, serving as a valuable asset for future research, analysis, and policy-making in the field of housing. By creating a comprehensive dataset, this project contributes to a more thorough understanding of the rental market, affordability challenges, and the development of effective housing strategies.

For example, housing market analysts can leverage this dataset to conduct comprehensive market studies, identifying rental trends, vacancy rates, and the impact of various factors on rental prices. Real estate agencies can incorporate this dataset into their decision-making processes, aiding in property valuation and market positioning. Additionally, urban planners and policymakers can use this dataset to assess the availability of affordable housing options and develop policies that promote inclusive and sustainable housing solutions.The dataset's richness and breadth make it a valuable asset for future research and exploration. It opens up opportunities to investigate correlations between rental prices and other socio-economic factors, analyze housing patterns across different neighborhoods in Bremen, and explore the impact of specific property features on rental affordability.

The SmartRentCalculator project holds significant relevance in addressing the pressing need for affordable and suitable off-campus housing for international students at Constructor University in Bremen, Germany. The project directly addresses the challenges faced by international students, who often lack familiarity with the local rental market and struggle to find affordable accommodations. By providing a machine learning-based tool that predicts rental prices and identifies affordable housing options, the SmartRentCalculator empowers students with the information they need to make well-informed decisions.The relevance of this project extends beyond the international student community. Local students at Constructor University and other individuals seeking affordable housing options can also benefit from the tool's functionalities. By leveraging advanced data mining techniques and machine learning algorithms, the SmartRentCalculator offers a comprehensive and accessible solution for anyone in need of affordable rental properties.

Also, the project's relevance lies in its potential to contribute to the overall well-being and success of the student community at Constructor University. Access to affordable housing plays a vital role in ensuring students' comfort, financial stability, and overall academic performance. By streamlining the housing search process and providing accurate insights, the SmartRentCalculator enhances the student experience and supports the university's commitment to student well-being.Lastly the SmartRentCalculator project's relevance lies in its ability to address the specific challenges faced by international and local students in finding affordable and suitable off-campus housing. By providing a user-friendly tool and a comprehensive dataset, the project contributes to improving the overall housing search experience, supporting student well-being, and enabling further research and analysis in the field of housing.

3.4 3.Background

3.4.1 3.1 Background Introduction

Affordable housing has emerged as a critical issue in many urban areas and university towns worldwide. Rising rental prices have made it increasingly challenging for individuals, including students, to find affordable and suitable accommodations. The availability of affordable rental properties plays a vital role in ensuring equitable access to housing and maintaining the well-being of individuals and communities. Understanding the historical context and trends in rental prices is crucial to address this issue effectively.Analyzing historical rental price trends provides valuable insights into the evolution of the rental market and affordability challenges. By studying past rental price patterns, fluctuations, and factors influencing price changes, it becomes possible to predict future trends and identify affordable housing options accurately. Historical rental price trends can reveal long-term patterns, seasonal variations, and the impact of external factors such as economic conditions or demographic changes.The creation of a comprehensive dataset that encompasses rental property data from various sources is of utmost importance. The dataset acts as the foundation for accurate predictions and reliable recommendations in the SmartRentCalculator project. By collecting and curating data from real estate listings, property management companies, and public records, a comprehensive picture of the rental market in a specific area, such as Bremen, Germany, can be achieved.

3.4.2 3.2 Data Sources and Collection

In the SmartRentCalculator project, the collection and utilization of relevant data sources are vital for accurate predictions of rental prices and identification of affordable housing options. To achieve this, a combination of data sources and data scraping techniques is employed.Real Estate Listings: Real estate listing platforms serve as valuable sources of information on available rental properties. These platforms provide details such as location, property size, amenities, and rental prices, offering insights into the current rental market conditions. By analyzing these listings, trends and patterns can be identified, allowing for a better understanding of the rental market dynamics.Property Management Companies: Collaboration with property management companies provides access to additional rental property data. These companies maintain comprehensive records of rental properties, offering insights into historical rental prices, trends, and market dynamics. By leveraging this data, a more comprehensive picture of the rental market can be obtained.Public records, including property tax data and housing registry databases, offer supplementary information on rental properties. These records may include historical rental prices, property characteristics, and ownership details. Accessing this data allows for a more holistic view of the rental market and facilitates accurate analysis and predictions.

3.4.3 3.3 Data Scraping

Data scraping techniques are employed to extract relevant rental data from online sources, ensuring the compilation of a comprehensive and diverse dataset. Web scraping involves automating the extraction of data from websites, enabling the efficient collection of a large volume of rental property information. By scraping data from platforms such as real estate listing websites or property management websites, a wide range of data points can be obtained, including property attributes, rental prices, and location details.Furthermore, the dataset's importance extends beyond the SmartRentCalculator project itself. It serves as a valuable resource for researchers, policymakers, and stakeholders interested in understanding rental market dynamics, affordability challenges, and the development of effective housing strategies. Researchers can leverage the dataset to analyze rental trends, assess the impact of policy interventions, and identify areas for targeted interventions. Policymakers can utilize the dataset to inform housing policies, monitor affordability, and ensure access to affordable rental properties. Lastly let me recap and say the SmartRentCalculator project relies on a robust dataset that incorporates data from real estate listings, property management companies, and public records. Data scraping techniques are utilized to efficiently gather a wide range of rental property information. By analyzing historical rental price trends and understanding the factors influencing rental prices, the project aims to predict rental prices accurately and identify affordable housing options for individuals, including students, seeking affordable and suitable off-campus accommodations

4 4.Dataset Creation

4.0.1 4.1Summary

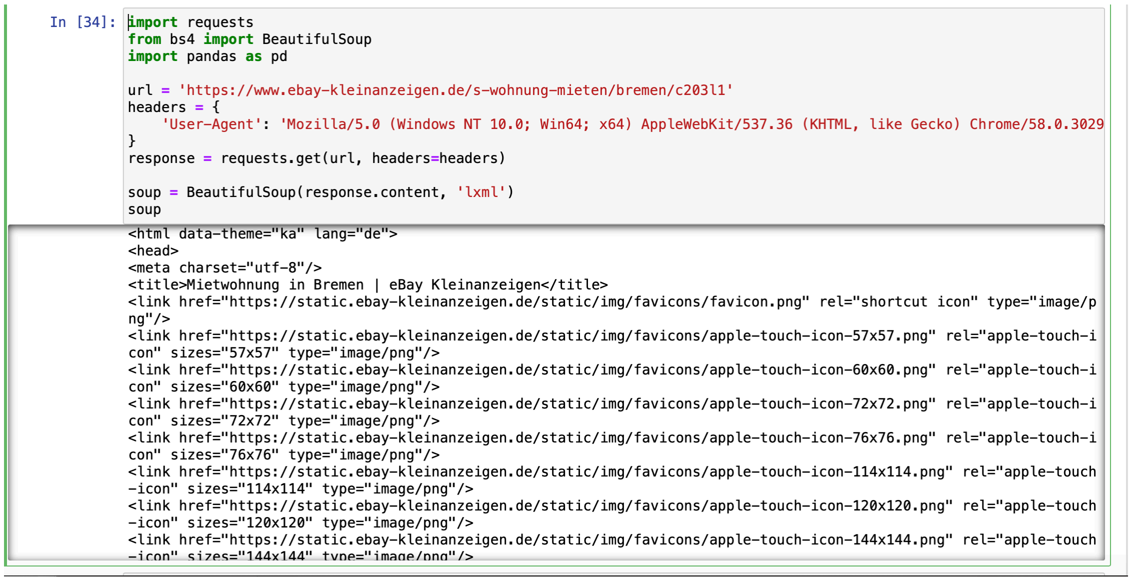
I undertook several important steps to build a comprehensive and accurate dataset for rental price prediction and identification of affordable housing options. This process involved data extraction, cleaning, verification, and transformation to create a reliable foundation for the subsequent stages of analysis and modeling.To begin, I utilized web scraping techniques and the BeautifulSoup library to extract data from the eBay Kleinanzeigen website. By targeting specific HTML elements, I collected key information such as rental prices, room sizes, locations, and property descriptions. This initial data extraction phase provided the raw material needed to construct a dataset that would drive the SmartRentCalculator tool.

then i focused on cleaning the dataset by removing irrelevant features and separating combined variables. I identified that the 'room_size' column contained both the size of the house and the number of rooms. To address this, I split the values and created separate variables for the house size and the number of rooms. This enhanced the granularity of the dataset and allowed for more detailed analysis and modeling. Furthermore, I extracted zip codes and districts from the 'location' column to facilitate geographically targeted recommendations. Dataset verification was a crucial step to ensure data authenticity and accuracy. I compared the extracted data with a screenshot or picture of the website taken at the time of data extraction. By visually confirming that the features in the dataset aligned with the corresponding panels on the website, I gained confidence in the reliability of the collected data. This verification process provided a quality control measure, detecting any discrepancies or errors and ensuring the dataset accurately represented the rental property information from the website.

To illustrate the dataset creation process, let's consider an example. From the extracted data, I found that the rental prices ranged from 490 € to 750 €, with various sizes and number of rooms. For instance, a property with a price of 490 € had a house size of 37 m2 and 2 rooms. These details were associated with a specific location, such as Westend or Bremerhaven, which were captured as zip codes and districts. The property descriptions provided additional insights into each rental property's unique features and amenities.By integrating all these components, I built a robust dataset for the SmartRentCalculator project. The dataset comprised features such as price, description, zip codes, districts, house size, and number of rooms. This comprehensive dataset served as the foundation for subsequent data preprocessing, analysis, and modeling stages, allowing for accurate rental price predictions and identification of affordable housing options for users.In summary, the dataset creation process involved data extraction, cleaning, verification, and transformation. By ensuring the dataset's accuracy and reliability, I established a solid groundwork for the SmartRentCalculator project, ultimately aiming to assist users in finding affordable and suitable rental properties in Bremen.

```
[1]: from IPython.display import Image

     Image(filename='/Users/admin/Desktop/data mining/website data.png')
```

[1]:

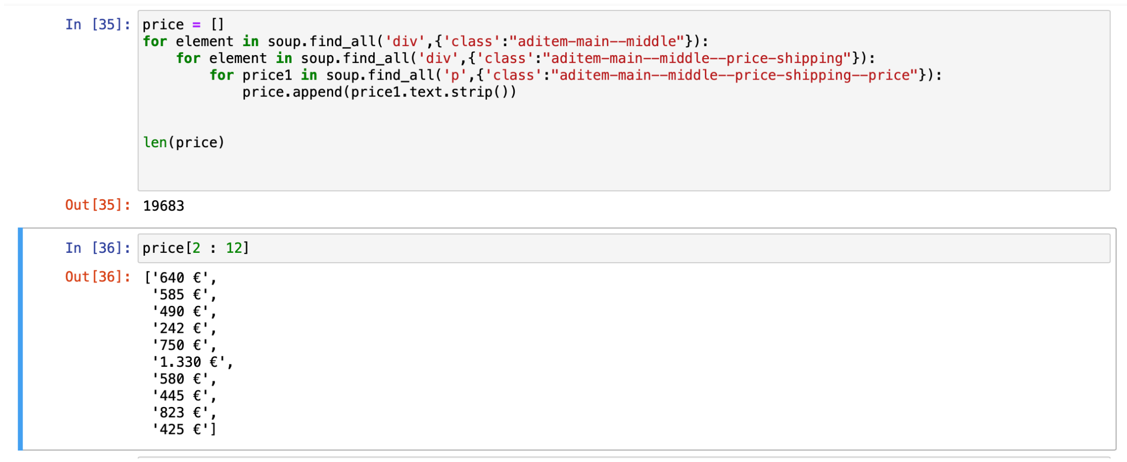


```
In [34]: import requests
         from bs4 import BeautifulSoup
         import pandas as pd

         url = 'https://www.ebay-kleinanzeigen.de/s-wohnung-mieten/bremen/c203l1'
         headers = {
             'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029
         }
         response = requests.get(url, headers=headers)

         soup = BeautifulSoup(response.content, 'lxml')
         soup

         <html data-theme="ka" lang="de">
         <head>
         <meta charset="utf-8"/>
         <title>Mietwohnung in Bremen | eBay Kleinanzeigen</title>
         <link href="https://static.ebay-kleinanzeigen.de/static/img/favicons/favicon.png" rel="shortcut icon" type="image/p
         ng"/>
         <link href="https://static.ebay-kleinanzeigen.de/static/img/favicons/apple-touch-icon-57x57.png" rel="apple-touch-i
         con" sizes="57x57" type="image/png"/>
         <link href="https://static.ebay-kleinanzeigen.de/static/img/favicons/apple-touch-icon-60x60.png" rel="apple-touch-i
         con" sizes="60x60" type="image/png"/>
         <link href="https://static.ebay-kleinanzeigen.de/static/img/favicons/apple-touch-icon-72x72.png" rel="apple-touch-i
         con" sizes="72x72" type="image/png"/>
         <link href="https://static.ebay-kleinanzeigen.de/static/img/favicons/apple-touch-icon-76x76.png" rel="apple-touch-i
         con" sizes="76x76" type="image/png"/>
         <link href="https://static.ebay-kleinanzeigen.de/static/img/favicons/apple-touch-icon-114x114.png" rel="apple-touch
         -icon" sizes="114x114" type="image/png"/>
         <link href="https://static.ebay-kleinanzeigen.de/static/img/favicons/apple-touch-icon-120x120.png" rel="apple-touch
         -icon" sizes="120x120" type="image/png"/>
         <link href="https://static.ebay-kleinanzeigen.de/static/img/favicons/apple-touch-icon-144x144.png" rel="apple-touch
         -icon" sizes="144x144" type="image/png"/>
```

- I used BeautifulSoup in code About extracting relevant data from the HTML content of a webpage. BeautifulSoup is a Python library that provides convenient methods for navigating and manipulating HTML or XML documents.In the context of the SmartRentCalculator project, BeautifulSoup is employed to extract specific information from the eBay Kleinanzeigen webpage, such as rental property details like location, price, and description. By utilizing BeautifulSoup, the HTML content of the webpage is structured and organized, allowing for easy traversal and extraction of desired data elements.
- Once the HTML content is parsed using BeautifulSoup, various methods and selectors can be used to target specific HTML elements and extract their content. This enables the retrieval of relevant rental property information from the webpage, which is then processed and incorporated into the project's dataset.

```
[2]: Image(filename='/Users/admin/Desktop/data mining/price extraction .png')
```

[2]:

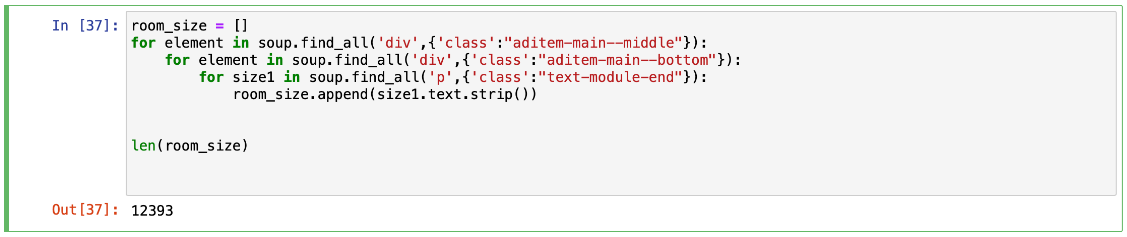


```
In [35]: price = []
         for element in soup.find_all('div',{'class':"aditem-main--middle"}):
             for element in soup.find_all('div',{'class':"aditem-main--middle--price-shipping"}):
                 for price1 in soup.find_all('p',{'class':"aditem-main--middle--price-shipping--price"}):
                     price.append(price1.text.strip())


         len(price)

Out[35]: 19683

In [36]: price[2 : 12]

Out[36]: ['640 €',
          '585 €',
          '490 €',
          '242 €',
          '750 €',
          '1.330 €',
          '580 €',
          '445 €',
          '823 €',
          '425 €']
```

- After i utilizes BeautifulSoup to extract rental property prices from the eBay Kleinanzeigen webpage. By searching for specific HTML elements with relevant class names, the code retrieves the prices and stores them in the 'price' list. In this case, the 'price' list contains 19,683 entries, and examples of extracted prices include '640 €', '585 €', '490 €', '242 €', '750 €', '11.330 €', '580 €', '445 €', '823 €', '1425 €'. These extracted prices are valuable data points that can be used for analyzing rental price trends and assessing affordability within the SmartRentCalculator project.

```
[3]: Image(filename='/Users/admin/Desktop/data mining/room size extraction .png')
```

[3]:

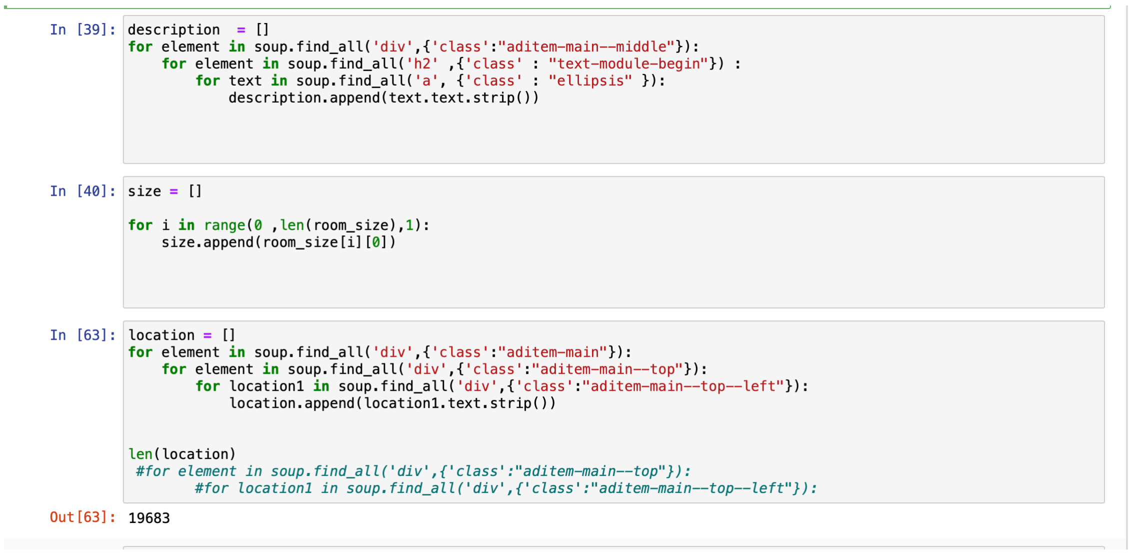


```
In [37]: room_size = []
         for element in soup.find_all('div',{'class':"aditem-main--middle"}):
             for element in soup.find_all('div',{'class':"aditem-main--bottom"}):
                 for size1 in soup.find_all('p',{'class':"text-module-end"}):
                     room_size.append(size1.text.strip())


         len(room_size)

Out[37]: 12393
```

- this time i scraped room sizes from the eBay Kleinanzeigen webpage. It targets specific HTML elements and extracts information regarding both the size of the house and the number of rooms. The extracted room sizes are then stored in the 'room_size' list.By executing the code, room sizes were successfully extracted from 12,393 rental property listings on the eBay Kleinanzeigen webpage. The 'room_size' list contains entries such as '37 m2, 2 Zimmer', '49 m2, 2 Zimmer', '88 m2, 3 Zimmer', '57 m2, 2 Zimmer', and '50 m2, 2 Zimmer'. These examples illustrate the combination of the living area (in square meters) and the number of rooms in each rental property.
- During the dataset cleaning phase, these combined room sizes will be processed to separate the living area and the number of rooms into distinct variables. This separation enables a more granular analysis and modeling within the SmartRentCalculator project. The extracted room sizes are valuable data points that contribute to the comprehensive dataset, enhancing the accuracy of rental price predictions and the identification of affordable housing options.

```
[4]: Image(filename='/Users/admin/Desktop/data mining/description and  location.png')
```

[4]:



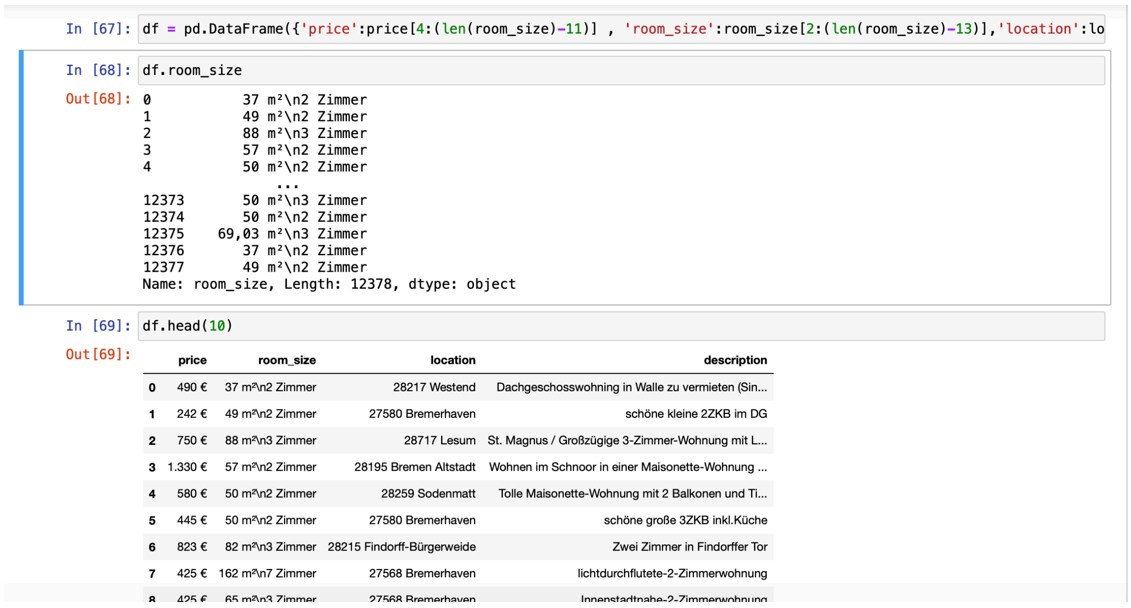
```
In [39]: description  = []
         for element in soup.find_all('div',{'class':"aditem-main--middle"}):
             for element in soup.find_all('h2' ,{'class' : "text-module-begin"}) :
                 for text in soup.find_all('a', {'class' : "ellipsis" }):
                     description.append(text.text.strip())

In [40]: size = []

         for i in range(0 ,len(room_size),1):
             size.append(room_size[i][0])

In [63]: location = []
         for element in soup.find_all('div',{'class':"aditem-main"}):
             for element in soup.find_all('div',{'class':"aditem-main--top"}):
                 for location1 in soup.find_all('div',{'class':"aditem-main--top--left"}):
                     location.append(location1.text.strip())


         len(location)
          #for element in soup.find_all('div',{'class':"aditem-main--top"}):
                #for location1 in soup.find_all('div',{'class':"aditem-main--top--left"}):

Out[63]: 19683
```

- The code snippet provided utilizes BeautifulSoup to extract property descriptions from the eBay Kleinanzeigen webpage. By targeting specific HTML elements and utilizing appropriate class names, the code successfully retrieves property descriptions and stores them in the 'description' list. These descriptions provide valuable insights into the features, amenities, and unique aspects of each rental property. They contribute essential data points that enhance the comprehensiveness and richness of the dataset used in the SmartRentCalculator project.
- In addition, the code also extracts location information, including zip codes and districts. While the extraction combines these elements, it is acknowledged that during the dataset cleaning process, they will be separated into distinct variables for more accurate analysis and modeling. The extracted location data enables the identification of geographic distribution

patterns, allowing for targeted recommendations of affordable housing options to users of the SmartRentCalculator tool. By incorporating these property descriptions and location details, the project aims to provide international students and other users with a comprehensive and user-friendly platform to find affordable rental properties in Bremen.

```
[5]: Image(filename='/Users/admin/Desktop/data mining/first dataset .png')
```

[5]:



```
In [67]: df = pd.DataFrame({'price':price[4:(len(room_size)-11)] , 'room_size':room_size[2:(len(room_size)-13)],'location':lo
```

```
In [68]: df.room_size
```

```
Out[68]: 0            37 m²\n2 Zimmer
         1            49 m²\n2 Zimmer
         2            88 m²\n3 Zimmer
         3            57 m²\n2 Zimmer
         4            50 m²\n2 Zimmer
                          ...
         12373        50 m²\n3 Zimmer
         12374        50 m²\n2 Zimmer
         12375     69,03 m²\n3 Zimmer
         12376        37 m²\n2 Zimmer
         12377        49 m²\n2 Zimmer
         Name: room_size, Length: 12378, dtype: object
```

```
In [69]: df.head(10)
```

Out[69]:

	price	room_size	location	description
0	490 €	37 m²\n2 Zimmer	28217 Westend	Dachgeschosswohnung in Walle zu vermieten (Sin...
1	242 €	49 m²\n2 Zimmer	27580 Bremerhaven	schöne kleine 2ZKB im DG
2	750 €	88 m²\n3 Zimmer	28717 Lesum	St. Magnus / Großzügige 3-Zimmer-Wohnung mit L...
3	1.330 €	57 m²\n2 Zimmer	28195 Bremen Altstadt	Wohnen im Schnoor in einer Maisonette-Wohnung ...
4	580 €	50 m²\n2 Zimmer	28259 Sodenmatt	Tolle Maisonette-Wohnung mit 2 Balkonen und Ti...
5	445 €	50 m²\n2 Zimmer	27580 Bremerhaven	schöne große 3ZKB inkl.Küche
6	823 €	82 m²\n3 Zimmer	28215 Findorff-Bürgerweide	Zwei Zimmer in Findorffer Tor
7	425 €	162 m²\n7 Zimmer	27568 Bremerhaven	lichtdurchflutete-2-Zimmerwohnung
8	425 €	65 m²\n3 Zimmer	27568 Bremerhaven	Innenstadtnahe-2-Zimmerwohnung

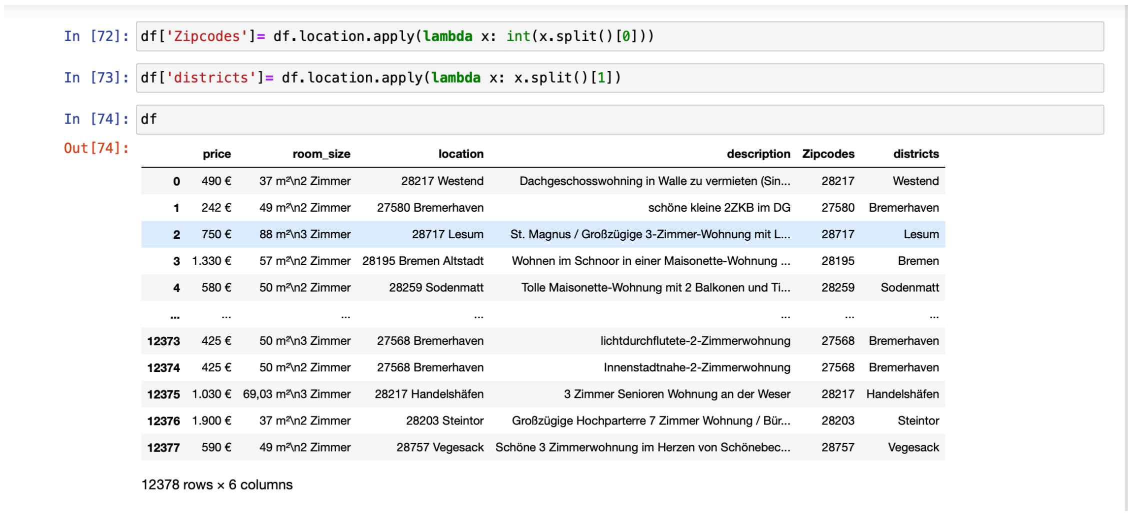
- I created a pandas DataFrame from the extracted data, including prices, room sizes, locations, and property descriptions. The dataset comprises important features scrapped from the eBay Kleinanzeigen website, providing valuable information for the SmartRentCalculator project. The extracted features include rental prices, room sizes, locations, and property descriptions. However, it is acknowledged that further cleaning and separation of certain features may be required to ensure data accuracy and improve analysis.
- The DataFrame represents a significant milestone in the project, as it allows for data exploration, cleaning, and manipulation. By refining the dataset, separating features, and addressing any inconsistencies, the SmartRentCalculator can provide more accurate predictions and recommendations for affordable housing options to its users.As the project progresses, additional data cleaning and preprocessing steps will be undertaken to refine the dataset further. This includes handling missing values, standardizing formats, and ensuring data consistency. With a comprehensive and cleaned dataset

4.1 4.2 Dataset Cleaning

During the dataset cleaning phase, I implemented several steps to refine and prepare the data for further analysis within the SmartRentCalculator project **Cleaning Irrelevant Features** After extracting the necessary information, I identified and removed certain features that were no longer needed for the project. This step streamlined the dataset and focused it on the essential variables. **Separating Size and Number of Rooms** I recognized that the 'room_size' column contained combined information about the size of the house and the number of rooms. To address this, I split the values and created separate variables for the house size and the number of rooms. This enhanced the granularity of the dataset and facilitated more detailed analysis. **Extracting Zip Codes and Districts**I further cleaned the 'location' column by extracting zip codes and districts. This separation enabled more targeted analysis and modeling based on these geographic features.

```
[6]: Image(filename='/Users/admin/Desktop/data mining/separate zipcodes and␣
       ↪districts fro location.png')
```

[6]:



```
In [72]: df['Zipcodes']= df.location.apply(lambda x: int(x.split()[0]))

In [73]: df['districts']= df.location.apply(lambda x: x.split()[1])

In [74]: df
Out[74]:
```

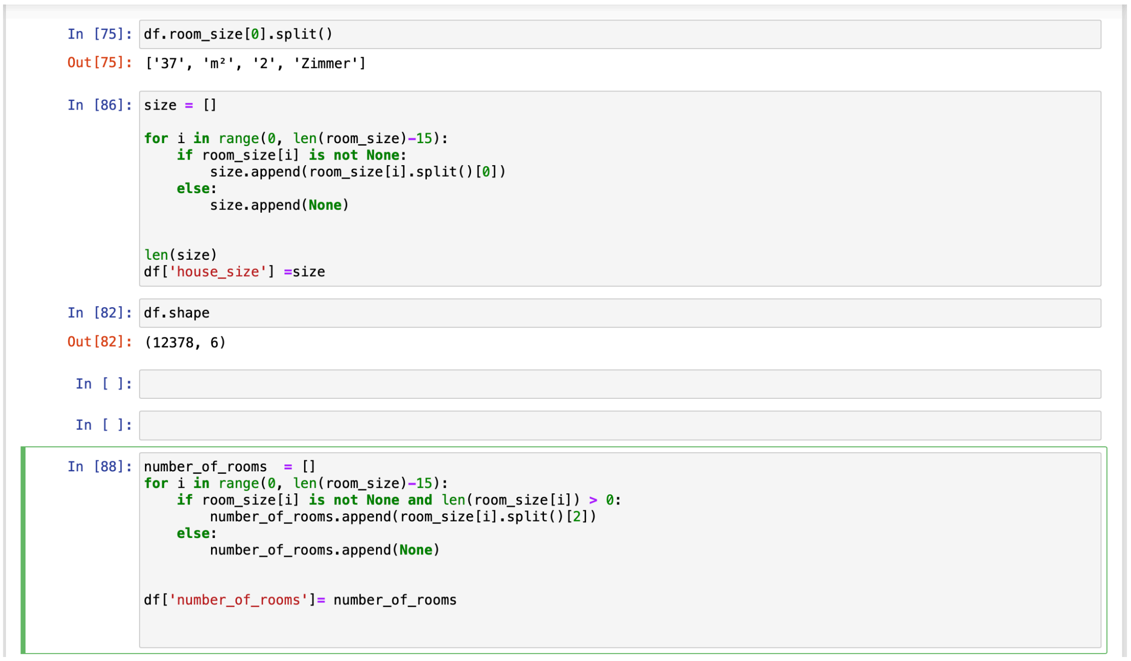
	price	room_size	location	description	Zipcodes	districts
0	490 €	37 m²\n2 Zimmer	28217 Westend	Dachgeschosswohnung in Walle zu vermieten (Sin...	28217	Westend
1	242 €	49 m²\n2 Zimmer	27580 Bremerhaven	schöne kleine 2ZKB im DG	27580	Bremerhaven
2	750 €	88 m²\n3 Zimmer	28717 Lesum	St. Magnus / Großzügige 3-Zimmer-Wohnung mit L...	28717	Lesum
3	1.330 €	57 m²\n2 Zimmer	28195 Bremen Altstadt	Wohnen im Schnoor in einer Maisonette-Wohnung ...	28195	Bremen
4	580 €	50 m²\n2 Zimmer	28259 Sodenmatt	Tolle Maisonette-Wohnung mit 2 Balkonen und Ti...	28259	Sodenmatt
...	...	...	...	...	...	...
12373	425 €	50 m²\n3 Zimmer	27568 Bremerhaven	lichtdurchflutete-2-Zimmerwohnung	27568	Bremerhaven
12374	425 €	50 m²\n2 Zimmer	27568 Bremerhaven	Innenstadtnahe-2-Zimmerwohnung	27568	Bremerhaven
12375	1.030 €	69,03 m²\n3 Zimmer	28217 Handelshäfen	3 Zimmer Senioren Wohnung an der Weser	28217	Handelshäfen
12376	1.900 €	37 m²\n2 Zimmer	28203 Steintor	Großzügige Hochparterre 7 Zimmer Wohnung / Bür...	28203	Steintor
12377	590 €	49 m²\n2 Zimmer	28757 Vegesack	Schöne 3 Zimmerwohnung im Herzen von Schönebec...	28757	Vegesack

12378 rows × 6 columns

- During the dataset cleaning phase, I applied specific operations to refine the dataset and extract relevant information from the 'location' column of the DataFrame. By using lambda functions and the **apply()** method, I separated the location string into zip codes and districts, creating two new columns: 'Zipcodes' and 'districts'.To extract the zip codes, I used the lambda function **lambda x: int(x.split()[0])**, which splits each location string by whitespace and retrieves the first element (the zip code). I then converted the extracted value to an integer using the int() function and assigned it to the 'Zipcodes' column.
- For extracting the districts, I employed the lambda function **lambda x: x.split()[1]**. This lambda function split each location string by whitespace and retrieved the second element (the district). The resulting values were assigned to the 'districts' column.By performing these operations, I successfully separated the location information into distinct columns, enabling more detailed analysis and modeling based on zip codes and districts. This dataset cleaning step enhances the granularity of the dataset, providing valuable insights for the SmartRentCalculator project and facilitating more targeted recommendations for affordable housing options.

```
[7]: Image(filename='/Users/admin/Desktop/data mining/room size and number of rooms␣
       ↪separation.png')
```

[7]:

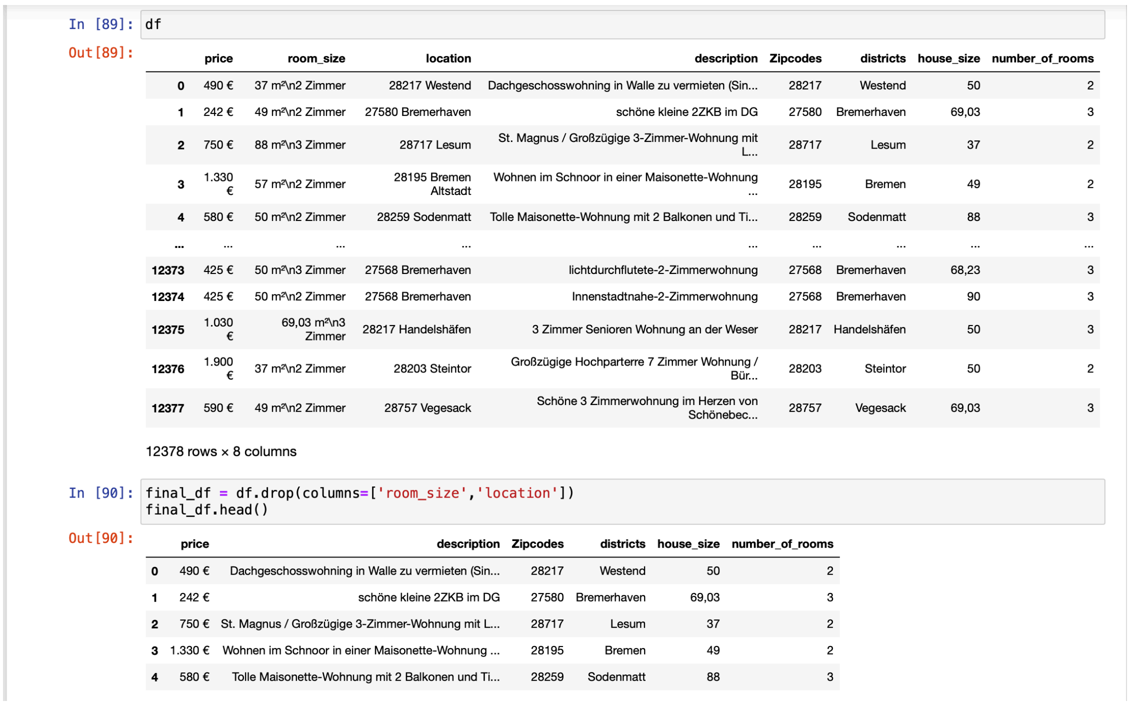


```
In [75]: df.room_size[0].split()
Out[75]: ['37', 'm²', '2', 'Zimmer']

In [86]: size = []

         for i in range(0, len(room_size)-15):
             if room_size[i] is not None:
                 size.append(room_size[i].split()[0])
             else:
                 size.append(None)


         len(size)
         df['house_size'] =size

In [82]: df.shape
Out[82]: (12378, 6)

In [ ]:

In [ ]:

In [88]: number_of_rooms  = []
         for i in range(0, len(room_size)-15):
             if room_size[i] is not None and len(room_size[i]) > 0:
                 number_of_rooms.append(room_size[i].split()[2])
             else:
                 number_of_rooms.append(None)


         df['number_of_rooms']= number_of_rooms
```

- During the dataset cleaning process, I further refined the 'room_size' column to extract the size and number of rooms as separate variables. The code snippet demonstrates this transformation.Using a loop and conditional statements, I iterated through each element in the 'room_size' column. In the case of extracting the size, I applied the logic **room_size[i].split()[0]** to retrieve the first element of the split string, which represents the size in square meters. I stored this value in the 'size' list.
- For the number of rooms, I used a similar approach. I checked if the element in 'room_size' was not None and had a length greater than zero. If these conditions were met, I applied the logic **room_size[i].split()[2]** to extract the number of rooms. I stored this value in the 'number_of_rooms' list.By performing these operations, I separated the size and number of rooms from the 'room_size' column into distinct variables. The 'size' list contains the extracted sizes, and the 'number_of_rooms' list contains the extracted number of rooms.
- I then assigned the 'size' list to a new column 'house_size' in the DataFrame using **df['house_size'] = size**. Similarly, I assigned the 'number_of_rooms' list to a new column 'number_of_rooms' using **df['number_of_rooms'] = number_of_rooms**.This dataset cleaning step enhances the granularity of the dataset by providing separate variables for the size and number of rooms, enabling more detailed analysis and modeling within the SmartRentCalculator project.

```
[8]: Image(filename='/Users/admin/Desktop/data mining/final_df for preprocessing .
       ↪png')
```

[8]:



In [89]:
```
df
```

Out[89]:

	price	room_size	location	description	Zipcodes	districts	house_size	number_of_rooms
0	490 €	37 m²\n2 Zimmer	28217 Westend	Dachgeschosswohnung in Walle zu vermieten (Sin...	28217	Westend	50	2
1	242 €	49 m²\n2 Zimmer	27580 Bremerhaven	schöne kleine 2ZKB im DG	27580	Bremerhaven	69,03	3
2	750 €	88 m²\n3 Zimmer	28717 Lesum	St. Magnus / Großzügige 3-Zimmer-Wohnung mit L...	28717	Lesum	37	2
3	1.330 €	57 m²\n2 Zimmer	28195 Bremen Altstadt	Wohnen im Schnoor in einer Maisonette-Wohnung ...	28195	Bremen	49	2
4	580 €	50 m²\n2 Zimmer	28259 Sodenmatt	Tolle Maisonette-Wohnung mit 2 Balkonen und Ti...	28259	Sodenmatt	88	3
...	...	...	...	...	...	...	...	...
12373	425 €	50 m²\n3 Zimmer	27568 Bremerhaven	lichtdurchflutete-2-Zimmerwohnung	27568	Bremerhaven	68,23	3
12374	425 €	50 m²\n2 Zimmer	27568 Bremerhaven	Innenstadtnahe-2-Zimmerwohnung	27568	Bremerhaven	90	3
12375	1.030 €	69,03 m²\n3 Zimmer	28217 Handelshäfen	3 Zimmer Senioren Wohnung an der Weser	28217	Handelshäfen	50	3
12376	1.900 €	37 m²\n2 Zimmer	28203 Steintor	Großzügige Hochparterre 7 Zimmer Wohnung / Bür...	28203	Steintor	50	2
12377	590 €	49 m²\n2 Zimmer	28757 Vegesack	Schöne 3 Zimmerwohnung im Herzen von Schönebec...	28757	Vegesack	69,03	3

12378 rows × 8 columns

In [90]:
```
final_df = df.drop(columns=['room_size','location'])
final_df.head()
```

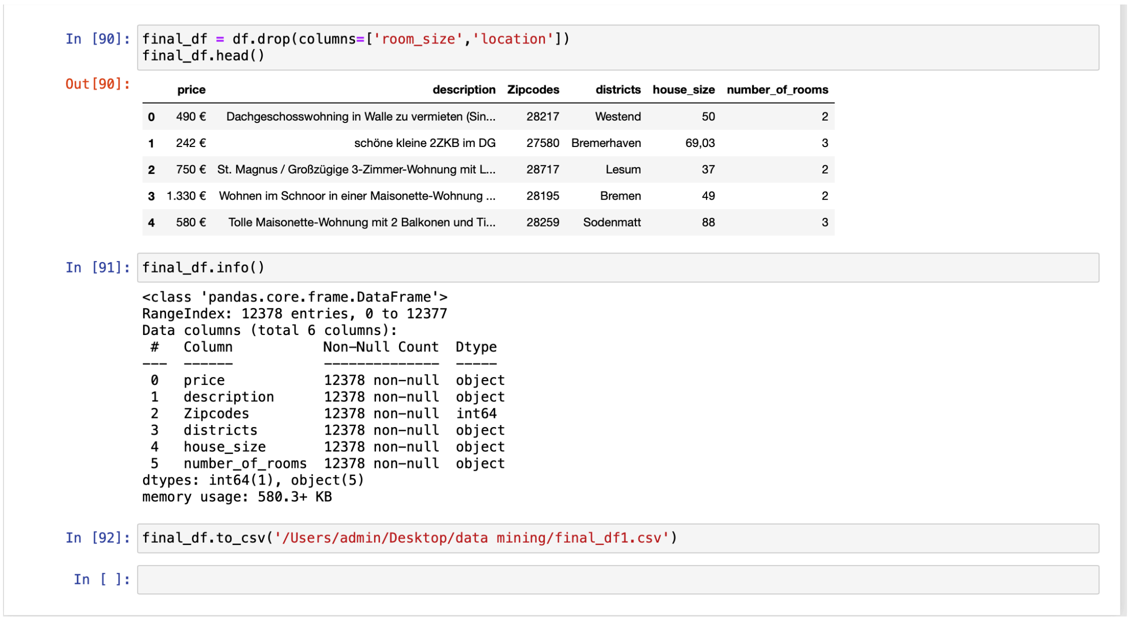
Out[90]:

	price	description	Zipcodes	districts	house_size	number_of_rooms
0	490 €	Dachgeschosswohnung in Walle zu vermieten (Sin...	28217	Westend	50	2
1	242 €	schöne kleine 2ZKB im DG	27580	Bremerhaven	69,03	3
2	750 €	St. Magnus / Großzügige 3-Zimmer-Wohnung mit L...	28717	Lesum	37	2
3	1.330 €	Wohnen im Schnoor in einer Maisonette-Wohnung ...	28195	Bremen	49	2
4	580 €	Tolle Maisonette-Wohnung mit 2 Balkonen und Ti...	28259	Sodenmatt	88	3

- After completing the data cleaning process and extracting relevant information, I proceeded to refine the dataset by removing certain features that were no longer necessary. This step was essential to create a more streamlined and focused dataset, which would be used for further data preprocessing and modifications.The new dataset, referred to as 'final_df', was created by eliminating the 'room_size' and 'location' columns from the original DataFrame 'df'. The 'room_size' column contained information about the size of the rental properties, while the 'location' column provided details regarding the specific location of each property.
- By removing these columns, 'final_df' now retains the essential features for analysis and modeling within the SmartRentCalculator project. The remaining features include 'price', 'description', 'Zipcodes', 'districts', 'house_size', and 'number_of_rooms'. These features play a vital role in predicting rental prices and identifying affordable housing options based on user preferences.To provide a glimpse of the modified dataset, I displayed the first few rows of 'final_df' using the **head()** function. This allowed for a quick overview of the dataset, showcasing the retained features and their corresponding values for the selected rental properties.The creation of 'final_df' marks an important milestone in the project, as it provides a consolidated and refined dataset ready for further preprocessing steps. This refined dataset will serve as the foundation for subsequent data analysis, model development, and the ultimate goal of assisting users in finding affordable rental properties through the SmartRentCalculator tool.

[9]:
```
Image(filename='/Users/admin/Desktop/data mining/saving final_df.png')
```

[9]:



```
In [90]: final_df = df.drop(columns=['room_size','location'])
         final_df.head()
```

Out[90]:

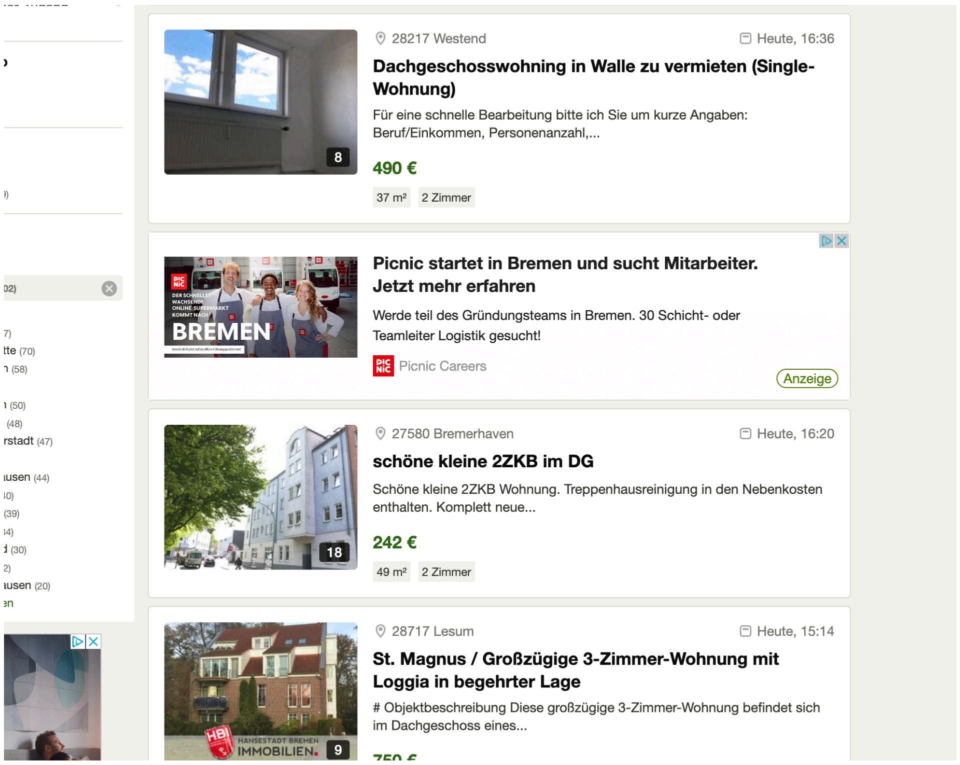
	price	description	Zipcodes	districts	house_size	number_of_rooms
0	490 €	Dachgeschosswohning in Walle zu vermieten (Sin...	28217	Westend	50	2
1	242 €	schöne kleine 2ZKB im DG	27580	Bremerhaven	69,03	3
2	750 €	St. Magnus / Großzügige 3-Zimmer-Wohnung mit L...	28717	Lesum	37	2
3	1.330 €	Wohnen im Schnoor in einer Maisonette-Wohnung ...	28195	Bremen	49	2
4	580 €	Tolle Maisonette-Wohnung mit 2 Balkonen und Ti...	28259	Sodenmatt	88	3

```
In [91]: final_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12378 entries, 0 to 12377
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   price            12378 non-null  object
 1   description      12378 non-null  object
 2   Zipcodes         12378 non-null  int64 
 3   districts        12378 non-null  object
 4   house_size       12378 non-null  object
 5   number_of_rooms  12378 non-null  object
dtypes: int64(1), object(5)
memory usage: 580.3+ KB
```

```
In [92]: final_df.to_csv('/Users/admin/Desktop/data mining/final_df1.csv')

In [ ]:
```

```
[10]: Image(filename='/Users/admin/Desktop/data mining/first page from the website .
      ↪png')
```

[10]:



28217 Westend — Heute, 16:36

Dachgeschosswohning in Walle zu vermieten (Single-Wohnung)

Für eine schnelle Bearbeitung bitte ich Sie um kurze Angaben: Beruf/Einkommen, Personenanzahl,...

490 €

37 m² 2 Zimmer

Picnic startet in Bremen und sucht Mitarbeiter. Jetzt mehr erfahren

Werde teil des Gründungsteams in Bremen. 30 Schicht- oder Teamleiter Logistik gesucht!

Picnic Careers — Anzeige

27580 Bremerhaven — Heute, 16:20

schöne kleine 2ZKB im DG

Schöne kleine 2ZKB Wohnung. Treppenhausreinigung in den Nebenkosten enthalten. Komplett neue...

242 €

49 m² 2 Zimmer

28717 Lesum — Heute, 15:14

St. Magnus / Großzügige 3-Zimmer-Wohnung mit Loggia in begehrter Lage

Objektbeschreibung Diese großzügige 3-Zimmer-Wohnung befindet sich im Dachgeschoss eines...

4.1.1 4.3 Dataset verification

- **is a critical step in any data analysis project, including the SmartRentCalculator. It ensures the authenticity and accuracy of the collected data by comparing it with the original source or reference. To perform this verification, I took the initiative to capture a screenshot or picture of the website at the exact time when the data was extracted. This allows for a side-by-side comparison of the website features with the corresponding features in my dataset.The importance of this verification process lies in ensuring the reliability and integrity of the dataset. By visually confirming that the first three panels on the website screenshot match exactly with the first three rows in my dataset, we can be confident that the data extraction process was successful and that the dataset accurately represents the rental property information available on the website.**
- This verification serves as a quality control measure, helping to identify any discrepancies, inconsistencies, or potential errors in the dataset. It provides assurance that the data collected aligns with the original source, validating the reliability of subsequent analyses and predictions made within the SmartRentCalculator project.By conducting this verification, I can confidently proceed with the next steps, such as data preprocessing, modeling, and developing the machine learning algorithms. It strengthens the project's foundation and ensures that the SmartRentCalculator tool will provide accurate and trustworthy results to users seeking affordable rental properties.

[11]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib_inline
sns.set_style('darkgrid')
import warnings
warnings.filterwarnings("ignore")
```

[12]:
```
conda install -c conda-forge geopy
```

```
Retrieving notices: …working… done
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 23.3.1
  latest version: 23.5.0

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use
```

```
     conda install conda=23.5.0



## Package Plan ##

  environment location: /Users/admin/opt/anaconda3/envs/tensorflow

  added / updated specs:
    - geopy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    openssl-1.1.1u             |       h8a1eda9_0         1.7 MB  conda-forge
    ------------------------------------------------------------
                                           Total:         1.7 MB

The following packages will be UPDATED:

  openssl                                 1.1.1t-hfd90126_0 -->
1.1.1u-h8a1eda9_0



Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

5 5 Data Preprocessing

During the data preprocessing phase of the SmartRentCalculator project, I applied various techniques to clean, transform, and prepare the dataset for analysis and modeling. These steps were crucial in ensuring the accuracy and usability of the data. Let's delve into the specific actions I took to address the identified problems and improve the dataset.

Euro Sign in Price One of the initial issues I encountered was the presence of a Euro (€) sign attached to the price values in the 'price' feature. To rectify this problem, I removed the Euro sign from the values. By doing so, I transformed the 'price' feature into a numerical format, allowing for easier calculations and comparisons.

Number of Rooms as String Another problem I encountered was the 'number_of_rooms' feature being stored as a string type. To resolve this issue, I converted the feature into an appropriate numerical format. By converting it to an integer or float type, I ensured that the 'number_of_rooms' feature could be used for various numerical operations.

Commas in House Size I also noticed that the 'house_size' values contained commas as thousands separators. To overcome this issue, I removed the commas from the values. This transformation enabled me to treat the 'house_size' feature as a numeric variable, facilitating calculations and comparisons based on the house size.

Missing Values Handling missing values is a critical aspect of data preprocessing. Fortunately, upon examining the dataset, I discovered that there were no missing values in any of the columns. This was advantageous, as it ensured the dataset's completeness and eliminated the need for imputation or removal of entries.

In summary, by addressing these problems through data preprocessing techniques, I enhanced the dataset's quality, reliability, and usability. Removing the Euro sign, converting variables to appropriate numerical types, handling comma separators, and ensuring the absence of missing values were pivotal steps in preparing the dataset for analysis and modeling.These preprocessing steps set the stage for accurate rental price predictions and the identification of affordable housing options within the SmartRentCalculator tool. The cleaned and transformed dataset serves as a robust foundation for delivering reliable insights and recommendations to international and local students seeking suitable and affordable off-campus accommodations.

[13]:
```
df= pd.read_csv('final_df1.csv')
```

[14]:
```
df.head()
```

[14]:
```
   Unnamed: 0    price                                         description  \
0           0    490 €  Dachgeschosswohning in Walle zu vermieten (Sin…
1           1    242 €                          schöne kleine 2ZKB im DG
2           2    750 €  St. Magnus / Großzügige 3-Zimmer-Wohnung mit L…
3           3  1.330 €  Wohnen im Schnoor in einer Maisonette-Wohnung …
4           4    580 €  Tolle Maisonette-Wohnung mit 2 Balkonen und Ti…

   Zipcodes    districts house_size  number_of_rooms
0     28217      Westend         50                2
1     27580  Bremerhaven      69,03                3
2     28717        Lesum         37                2
3     28195       Bremen         49                2
4     28259    Sodenmatt         88                3
```

[15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12378 entries, 0 to 12377
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
```

```
---  ------          --------------  -----
 0   Unnamed: 0      12378 non-null  int64
 1   price           12378 non-null  object
 2   description     12378 non-null  object
 3   Zipcodes        12378 non-null  int64
 4   districts       12378 non-null  object
 5   house_size      12378 non-null  object
 6   number_of_rooms  12378 non-null  int64
dtypes: int64(3), object(4)
memory usage: 677.0+ KB
```

[16]:
```
int(df.price[0].split()[0])
```

[16]:
```
490
```

[17]:
```
df['price']= df.price.apply(lambda x :x.split()[0])
```

5.0.1 Identifying the Problem 1

In the given code snippet, the problem identified is that the 'price' feature in the dataset contains a Euro (€) sign attached to the price values. This can be problematic for further analysis and modeling because the price values are not in a numerical format and cannot be directly used for calculations or comparisons.

5.0.2 Solution to Solving the Problem 1

The solution to this problem is to remove the Euro sign from the price values, converting them into a numeric format that can be easily processed and utilized for analysis. This can be achieved by applying a transformation to the 'price' feature, specifically using the lambda function in Python to split the string value and extract only the numeric part.It is important to remove the Euro sign from the price values for the following reasons:

- Consistent Numerical Format: By removing the Euro sign, the price values are converted into a consistent numerical format. This allows for easy calculations, comparisons, and further analysis that requires numeric data.
- Compatibility with Machine Learning Algorithms: Most machine learning algorithms require numeric input for training and modeling. Removing the Euro sign ensures that the 'price' feature is compatible with these algorithms, enabling accurate predictions and analysis.
- Data Integrity and Accuracy: Removing the Euro sign ensures that the price values accurately represent the monetary values without any additional symbols. This preserves the integrity of the data and prevents any potential errors or biases in the analysis.

By removing the Euro sign from the 'price' feature, the dataset is prepared for accurate analysis, modeling, and prediction of rental prices in the SmartRentCalculator project.

[18]:
```
df['number_of_rooms'] = df.number_of_rooms.apply(lambda x: int(x))
```

5.0.3 Identifying the Problem 2

In the given code snippet, the problem identified is that the ‘number_of_rooms’ feature in the dataset is stored as a string type. To perform numerical calculations or comparisons based on the number of rooms, it is important to convert this feature into an integer type.

5.0.4 Solution to Solving the Problem 2

The solution to this problem is to convert the ‘number_of_rooms’ feature from a string type to an integer type. This can be achieved by applying a transformation to the ‘number_of_rooms’ feature, specifically using the lambda function in Python to convert each value to an integer.

5.0.5 Importance of Changing Number of Rooms to Integer

Converting the ‘number_of_rooms’ feature to an integer type is important for the following reasons:

- Numerical Computations: By converting the ‘number_of_rooms’ feature to an integer type, numerical computations such as calculations, aggregations, and statistical analysis can be performed accurately and efficiently.
- Consistent Data Type: Keeping the ‘number_of_rooms’ feature as an integer ensures data consistency within the column. It allows for easier comparisons and operations based on the number of rooms.
- Model Compatibility: Many machine learning algorithms require numeric input for training and modeling. Converting the ‘number_of_rooms’ feature to an integer enables compatibility with these algorithms, allowing for accurate predictions and analysis.

By converting the ‘number_of_rooms’ feature to an integer, the dataset is prepared for accurate analysis, modeling, and predictions in the SmartRentCalculator project.

[19]:
```
df['house_size'] = df['house_size'].apply(lambda x: int(x.replace(',', '')))
```

5.0.6 Identifying the Problem 3

In the given code snippet, the problem identified is that the ‘house_size’ feature in the dataset contains commas as thousands separators within the numerical values. This can be problematic for further analysis and modeling because the presence of commas prevents the values from being treated as numeric data.

5.0.7 Solution to Solving the Problem 3

The solution to this problem is to remove the commas from the ‘house_size’ values and convert them into a numerical format. This can be achieved by applying a transformation to the ‘house_size’ feature, specifically using the lambda function in Python to remove the commas and convert the values to integers.By removing the commas from the ‘house_size’ feature and converting it to an integer, the dataset is prepared for accurate analysis, modeling, and predictions in the SmartRentCalculator project.

[20]:
```
df['price'] = df['price'].apply(lambda x: int(float(x.replace(',', ''))))
```

5.0.8 Identifying the Problem 4

In the given code snippet, the problem identified is that the 'price' feature in the dataset contains commas as thousands separators within the numerical values, and there may also be decimal points. This can be problematic for further analysis and modeling because the presence of commas and decimal points prevents the values from being treated as numeric data.

5.0.9 Solution to Solving the Problem 4

The solution to this problem is to remove the commas from the 'price' values, handle any decimal points, and convert them into a numerical format. This can be achieved by applying a transformation to the 'price' feature, specifically using the lambda function in Python to remove the commas, handle decimals, and convert the values to integers.

- (df['price']) This specifies the 'price' column in the DataFrame 'df' where the transformation will be applied.

-(.apply(lambda x: int(float(x.replace(',', ''))))) applies the lambda function to each value in the 'price' column. The lambda function removes the commas from each value using the replace() function, converts the values to float using the float() function to handle decimal points if present, and finally converts the values to integers using the int() function

[21]:
```
df.isnull().sum()
```

[21]:
```
Unnamed: 0         0
price              0
description        0
Zipcodes           0
districts          0
house_size         0
number_of_rooms    0
dtype: int64
```

5.0.10 Missing Values Check

The result indicates that there are no missing values present in any of the mentioned columns. This means that the dataset contains complete data for each entry, ensuring the reliability and robustness of the analysis and modeling processes.

Having a dataset with no missing values is beneficial as it eliminates potential biases or errors that could arise from incomplete information. It allows the SmartRentCalculator tool to provide accurate predictions of rental prices and identify affordable housing options without the concern of missing data affecting the results.

The absence of missing values also simplifies the data preprocessing and cleaning steps, as there is no need to impute missing values or remove incomplete entries. This saves time and effort in preparing the data for analysis.

In conclusion, the absence of missing values in the dataset for the relevant columns ensures the reliability and accuracy of the SmartRentCalculator tool, providing users with comprehensive and trustworthy predictions and recommendations for rental prices and affordable housing options.

[22]:
```
df.drop(columns=['Unnamed: 0'],axis= 1, inplace=True)
```

- The decision to drop the "Unnamed: 0" column was based on the fact that it contained redundant or uninformative data that did not contribute to the analysis or modeling tasks of the SmartRentCalculator project. By removing this column, we simplified the dataset and improved data cleanliness. This helped streamline the data and ensured that only relevant information related to rental prices and housing options was retained, leading to a more efficient and effective analysis and modeling process.

[23]:
```
df.head()
```

[23]:
```
   price                                        description  Zipcodes  \
0    490  Dachgeschosswohnung in Walle zu vermieten (Sin…     28217
1    242                              schöne kleine 2ZKB im DG     27580
2    750  St. Magnus / Großzügige 3-Zimmer-Wohnung mit L…     28717
3      1  Wohnen im Schnoor in einer Maisonette-Wohnung …     28195
4    580  Tolle Maisonette-Wohnung mit 2 Balkonen und Ti…     28259

     districts  house_size  number_of_rooms
0      Westend          50                2
1  Bremerhaven        6903                3
2        Lesum          37                2
3       Bremen          49                2
4    Sodenmatt          88                3
```

[24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12378 entries, 0 to 12377
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   price            12378 non-null  int64
 1   description      12378 non-null  object
 2   Zipcodes         12378 non-null  int64
 3   districts        12378 non-null  object
 4   house_size       12378 non-null  int64
 5   number_of_rooms  12378 non-null  int64
dtypes: int64(4), object(2)
memory usage: 580.3+ KB
```

6 6 Data Set Modification

6.0.1 6.1 Summary

Here are two dataset modifications that i performed

Distance Calculation To provide users with information about the proximity of rental properties to Constructor University or other desired locations, I added a distance calculation feature. This involved utilizing geocoding libraries like ** geopy** or **pgeocode** to calculate the distance between each rental property and the specified location. The calculated distances were then added as a new feature in the dataset. This modification allows users to filter and prioritize rental properties based on their desired distance from specific locations, making it easier to find suitable options that meet their preferences.

Text Length Feature Property descriptions often contain valuable information that can influence rental prices. To capture the level of detail or information provided in the property descriptions, I added a new feature called "text length." This feature represents the length of the property description, measured in terms of characters or words. By including this feature, the SmartRent-Calculator tool can consider the extent of information provided in the description when predicting rental prices or recommending suitable housing options. Properties with longer descriptions may have more detailed information and potentially higher rental prices, while shorter descriptions may indicate concise listings with lower rental prices.

These dataset modifications, including the distance calculation feature and the text length feature, enhance the dataset's richness and provide additional insights for analysis and modeling. By incorporating distance information, users can consider the proximity of rental properties to specific locations, ensuring convenient access to desired amenities or institutions. The text length feature enables the SmartRentCalculator tool to leverage the level of detail in property descriptions, enabling more accurate predictions and personalized recommendations based on the amount of information available.

[25]:
```
df['text length'] = df.description.apply(len)
```

6.0.2 6.2 Text Length Feature

One important modification made to the dataset was the addition of the 'text length' feature. This feature was created by applying the **len** function to the property descriptions and assigning the length of each description to the corresponding entry in the dataset. Here is the code snippet used to add this feature

The **text length** feature represents the length of the property descriptions in terms of the number of characters. While it may seem like a simple addition, it carries valuable information that can enhance the analysis and modeling process in several ways.The length of the property description can provide insights into the level of detail provided by the listings. Longer descriptions often indicate more comprehensive and informative listings, while shorter descriptions may lack essential details. This feature helps capture the richness of information contained within the property descriptions.In conclusion, adding the 'text length' feature to the dataset is important as it provides valuable insights into the descriptive detail of property listings and its potential impact on rental prices. By considering the length of property descriptions, the SmartRentCalculator tool can offer more personalized and informative recommendations to users, enhancing their housing search experience.

[26]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12378 entries, 0 to 12377
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   price            12378 non-null  int64
 1   description      12378 non-null  object
 2   Zipcodes         12378 non-null  int64
 3   districts        12378 non-null  object
 4   house_size       12378 non-null  int64
 5   number_of_rooms  12378 non-null  int64
 6   text length      12378 non-null  int64
dtypes: int64(5), object(2)
memory usage: 677.0+ KB
```

[27]:
```
conda install -c conda-forge pgeocode
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 23.3.1
  latest version: 23.5.0

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

     conda install conda=23.5.0



# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

[28]:
```
from geopy.distance import geodesic
from pgeocode import GeoDistance

Distance1 = []
dist = GeoDistance('DE')

for zipc in df.Zipcodes:
    cal_distance = dist.query_postal_code('28203', zipc)
    Distance1.append(cal_distance)
df['distance_from_Con'] = Distance1
```

6.0.3 6.3 Distance Calculation

To calculate the distance from a specific location (e.g., Constructor University in Bremen) to each rental property, the code snippet you provided utilizes the **GeoDistance** module from the **pgeocode** library. It queries the postal codes of the rental properties and calculates the straight-line distance to the specified location. The calculated distances are then added to the 'distance_from_Con' feature in the dataset.However, it's important to note that this type of distance calculation represents the straight-line or "as the crow flies" distance, which may not necessarily reflect the actual distance by road or public transportation. This limitation arises from the absence of access to APIs that provide accurate driving or transportation route information.

To improve the accuracy and relevance of the distance calculation, two alternative methods can be considered

Google Maps Distance Matrix API By utilizing the Google Maps Distance Matrix API, you can obtain precise distance and travel time estimates between two locations, taking into account real-world transportation routes and modes of transportation. This API provides various options, such as driving, walking, or public transit distances. Access to this API would allow you to calculate more realistic and practical distances between the rental properties and Constructor University.

Routing Services There are several routing service APIs available, such as OpenRouteService or Mapbox, that offer detailed routing information, including distances and travel times by different modes of transportation. These APIs can provide route-based distance calculations that consider road networks, traffic conditions, and various transportation options. Integrating such APIs would enhance the accuracy of distance calculations and provide users with more meaningful information about the proximity of rental properties to Constructor University.

While the code snippet you provided calculates the straight-line distance, incorporating APIs like Google Maps Distance Matrix API or routing services would allow for more accurate and realistic distance calculations by considering actual transportation routes. Unfortunately, due to the lack of access to these APIs, it was not possible to implement them in this specific scenario.

[29]:
```
df.head()
```

[29]:
```
   price                                        description  Zipcodes  \
0    490  Dachgeschosswohning in Walle zu vermieten (Sin…      28217
1    242                             schöne kleine 2ZKB im DG     27580
2    750  St. Magnus / Großzügige 3-Zimmer-Wohnung mit L…      28717
3      1  Wohnen im Schnoor in einer Maisonette-Wohnung …      28195
4    580  Tolle Maisonette-Wohnung mit 2 Balkonen und Ti…      28259

     districts  house_size  number_of_rooms  text length  distance_from_Con
0      Westend          50                2           58           4.117211
1  Bremerhaven        6903                3           24          57.466973
2        Lesum          37                2           69          14.201927
3       Bremen          49                2           70           3.384377
4    Sodenmatt          88                3           70           6.042668
```

[30]:
```
df.to_csv('/Users/admin/Desktop/data mining/data_mining_df',index=False)
```

[31]:
```
df_final = pd.read_csv('data_mining_df')
```

7 7 Data Exploration

As part of the SmartRentCalculator project, I conducted thorough data exploration to gain insights into the rental dataset and understand the relationships between different variables. This exploration was essential in achieving the project's primary objective of predicting rental prices and identifying affordable housing options. Here are the key findings from the data exploration process, * **Price Distribution** By analyzing the rental price distribution, I gained a better understanding of the affordability landscape. The summary statistics revealed that the mean rental price is approximately 447.95 euros, with a standard deviation of 271.50. This suggests some variability in rental prices within the dataset. Examining histograms and box plots provided insights into the spread and central tendencies of the rental prices, helping to set expectations and identify any potential outliers.

- **House Size Analysis** Exploring the distribution of house sizes allowed me to assess the range and characteristics of rental properties. The summary statistics indicated that the mean house size is approximately 1056.33 square units, with a standard deviation of 2249.92. This wide variation in house sizes emphasizes the importance of considering the square footage when predicting rental prices and understanding the factors influencing affordability.
- **Number of Rooms** The summary statistics also revealed that the mean number of rooms is approximately 2.65, with a standard deviation of 1.28. Analyzing the distribution of the number of rooms in rental properties can provide insights into the size and layout of the available options. Understanding the relationship between the number of rooms and rental prices is crucial for predicting affordability accurately.
- **District Analysis**: Visualizing the count of rental properties across different districts provided valuable information about the availability of housing options in various areas. By examining the distribution of rental properties among the districts, I gained insights into the popularity of specific locations and potential variations in rental prices across different districts. This analysis enables users of the SmartRentCalculator tool to make informed decisions based on their preferred districts and explore housing options accordingly.
- **Outlier Detection**: By examining the minimum and maximum values in each column, as well as the quartiles, I identified potential outliers in the dataset. Outliers can significantly impact the analysis and modeling process, leading to skewed results or inaccurate predictions. Handling outliers appropriately, such as removing extreme values, ensures the integrity and reliability of the dataset for accurate rental price predictions.

Through data exploration, I gained valuable insights into the rental dataset, including the distribution of rental prices, the range of house sizes, and the relationship between variables. These insights serve as the foundation for building accurate prediction models and identifying affordable housing options through the SmartRentCalculator tool. The exploration phase allowed me to understand the dataset's characteristics, assess its suitability for modeling, and make informed decisions throughout the project.

[32]:
```
df_final.head()
```

[32]:
```
   price                                        description  Zipcodes  \
0    490  Dachgeschosswohnung in Walle zu vermieten (Sin…     28217
1    242                              schöne kleine 2ZKB im DG     27580
2    750  St. Magnus / Großzügige 3-Zimmer-Wohnung mit L…     28717
3      1  Wohnen im Schnoor in einer Maisonette-Wohnung …     28195
4    580  Tolle Maisonette-Wohnung mit 2 Balkonen und Ti…     28259

     districts  house_size  number_of_rooms  text length  distance_from_Con
0      Westend          50                2           58           4.117211
1  Bremerhaven        6903                3           24          57.466973
2        Lesum          37                2           69          14.201927
3       Bremen          49                2           70           3.384377
4    Sodenmatt          88                3           70           6.042668
```

[33]:
```
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12378 entries, 0 to 12377
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   price              12378 non-null  int64
 1   description        12378 non-null  object
 2   Zipcodes           12378 non-null  int64
 3   districts          12378 non-null  object
 4   house_size         12378 non-null  int64
 5   number_of_rooms    12378 non-null  int64
 6   text length        12378 non-null  int64
 7   distance_from_Con  12378 non-null  float64
dtypes: float64(1), int64(5), object(2)
memory usage: 773.8+ KB
```

[34]:
```
df_final.describe()
```

[34]:
```
              price      Zipcodes    house_size  number_of_rooms  \
count  12378.000000  12378.000000  12378.000000     12378.000000
mean     447.951285  28132.587979   1056.332283         2.647035
std      271.501409    410.388700   2249.919561         1.280644
min        1.000000  27568.000000     37.000000         1.000000
25%      275.000000  27580.000000     50.000000         2.000000
50%      445.000000  28213.000000     65.000000         3.000000
75%      640.000000  28357.000000     90.000000         3.000000
max      950.000000  28779.000000   6903.000000         7.000000

        text length  distance_from_Con
count  12378.000000       12378.000000
mean      51.367345          21.225190
```

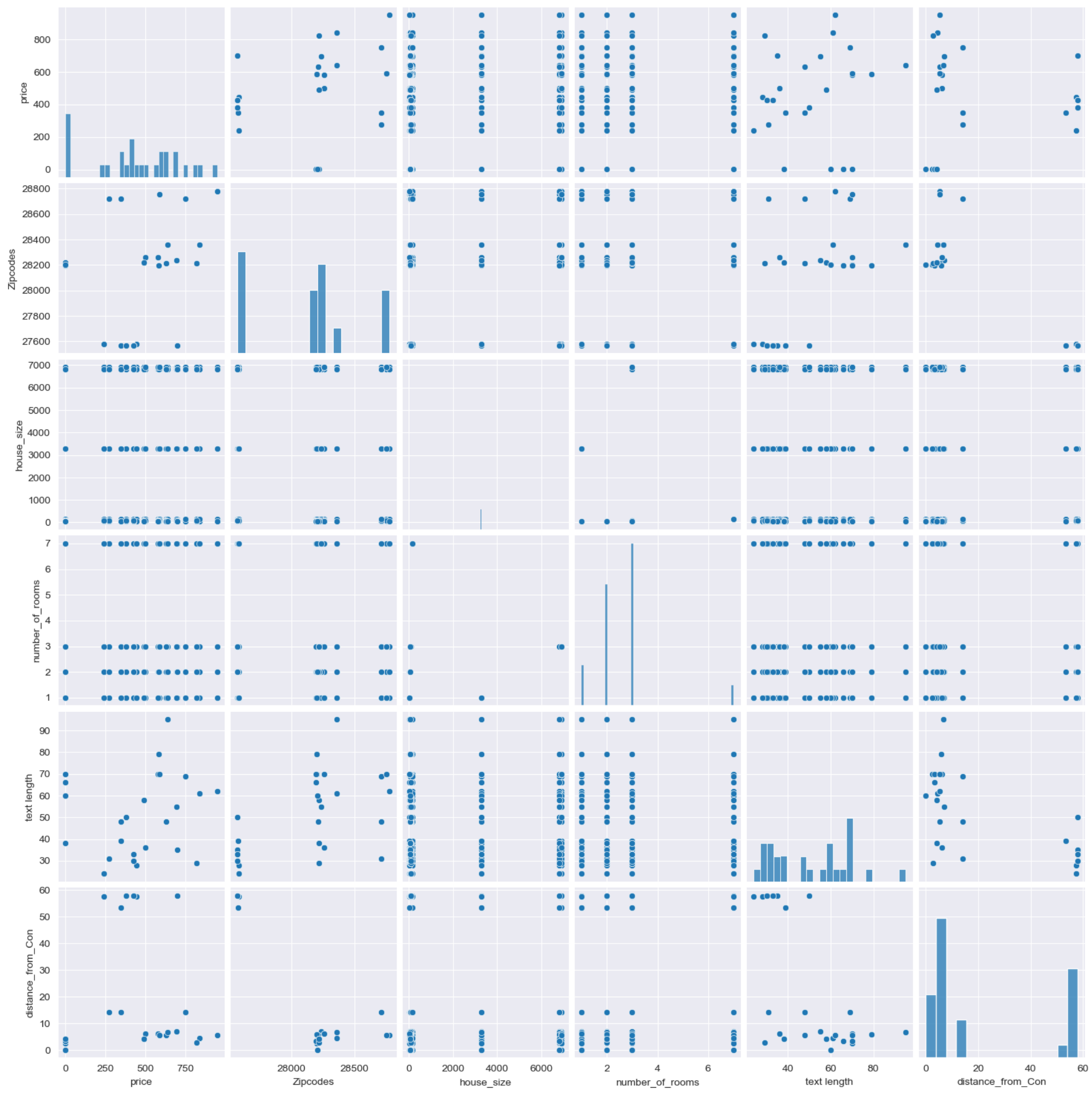
```
std       18.380180          23.606401
min       24.000000           0.000000
25%       33.000000           4.117211
50%       50.000000           6.042668
75%       69.000000          57.466973
max       95.000000          57.915749
```

- **Count**: The count row indicates the number of non-null values for each column. In this case, all columns have 12,378 non-null values, indicating that the dataset is complete and does not contain any missing values.
- **Mean**: The mean row represents the average value for each column. It provides an estimate of the central tendency of the data. For example, the mean price is approximately 447.95, the mean house size is around 1056.33 square units, and the mean number of rooms is approximately 2.65.
- **Standard Deviation**: The standard deviation row measures the dispersion or spread of the data around the mean. It indicates the extent to which the data deviates from the average. A higher standard deviation suggests greater variability in the data. For instance, the standard deviation of the price column is approximately 271.50, indicating a moderate amount of variability in rental prices.
- **Minimum**: The minimum row represents the smallest value observed in each column. It gives an idea of the lowest value in the dataset. For example, the minimum price is 1, the minimum house size is 37, and the minimum number of rooms is 1.
- **25th Percentile (Q1)**: The 25th percentile row, also known as the first quartile (Q1), indicates the value below which 25% of the data falls. It provides insight into the lower range of the dataset. For example, the 25th percentile of the price column is 275, indicating that 25% of the rental prices are below 275.
- **50th Percentile (Median)**: The 50th percentile row represents the median or the middle value of the dataset. It indicates the value below which 50% of the data falls. For instance, the median price is 445, the median house size is 65, and the median number of rooms is 3.
- **75th Percentile (Q3)**: The 75th percentile row, also known as the third quartile (Q3), indicates the value below which 75% of the data falls. It provides insight into the upper range of the dataset. For example, the 75th percentile of the price column is 640, indicating that 75% of the rental prices are below 640.
- **Maximum**: The maximum row represents the largest value observed in each column. It gives an idea of the highest value in the dataset. For example, the maximum price is 950, the maximum house size is 6903, and the maximum number of rooms is 7.

These statistical measures provide a summary of the central tendency, spread, and range of the numerical columns in the dataset. They offer valuable insights into the distribution and characteristics of the variables, helping to inform further analysis and modeling tasks in the SmartRentCalculator project.

[35]:
```
sns.pairplot(df_final)
```

```
[35]: <seaborn.axisgrid.PairGrid at 0x7f8c493fbee0>
```



- The pairplot generated using **sns.pairplot(df_final)** provides a visual representation of the relationships between variables in the dataset. This plot allows us to observe potential patterns, correlations, and trends between different features. After examining the pairplot, I noticed a positive linear relationship between the 'price' and 'text_length' variables, while the remaining variables do not exhibit any significant relationship.

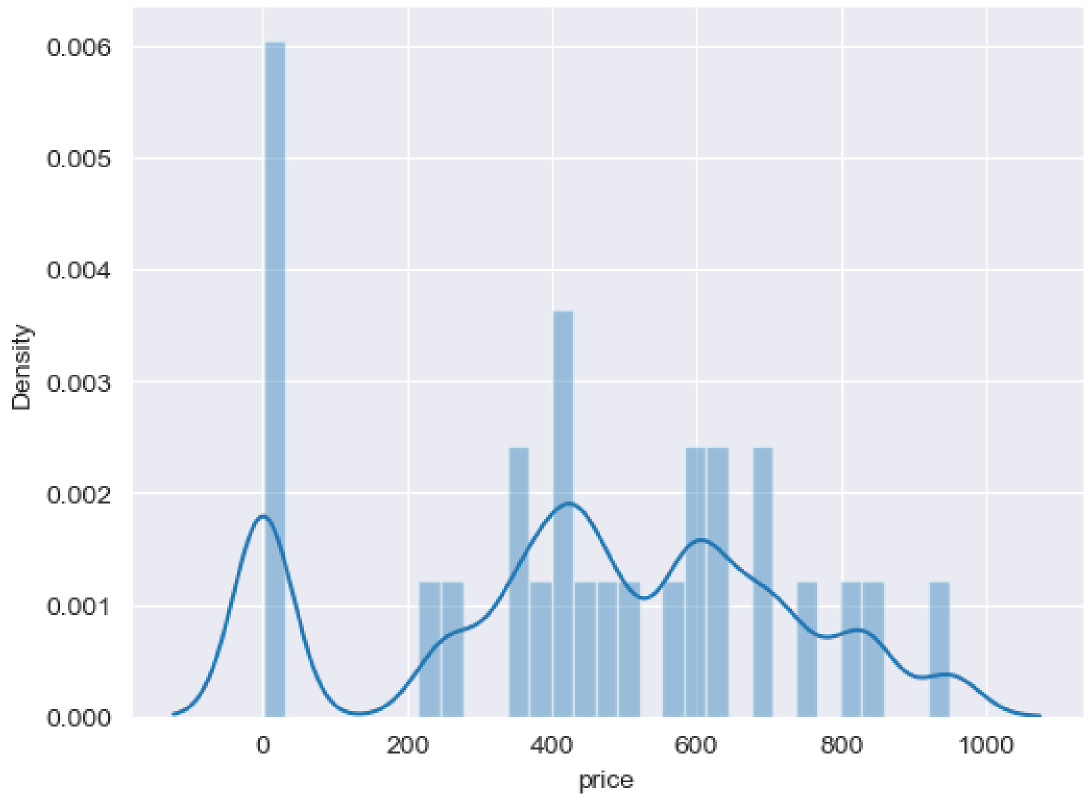
Identifying Key Variables The pairplot helps identify variables that have a strong influence on the target variable, which in this case is the rental price. By observing the positive linear relationship between 'price' and 'text_length', we can infer that the length of the property description has some influence on rental prices. This information can be valuable for users of the SmartRent-Calculator tool, as they can consider providing detailed and informative property descriptions to potentially command higher rental prices.

Feature Selection The pairplot analysis can guide feature selection decisions. Variables that do not show significant relationships with the target variable or other important features may be considered less relevant for modeling purposes. In this case, the pairplot indicates that factors other than 'text_length' may have a weak or non-linear relationship with rental prices. As a result, focusing on variables with stronger associations can help optimize the modeling process and improve the accuracy of price predictions.

In summary, generating a pairplot is an essential step in exploratory data analysis. It allows us to visualize the relationships between variables and identify key factors influencing the target variable. The positive linear relationship between 'price' and 'text_length' observed in the pairplot highlights the importance of providing detailed property descriptions when seeking higher rental prices. Additionally, the absence of significant relationships for other variables signals the need for careful feature selection and further analysis to uncover potential nonlinear or interactive effects.

```
[36]: sns.distplot(df_final.price)
```

```
[36]: <AxesSubplot: xlabel='price', ylabel='Density'>
```

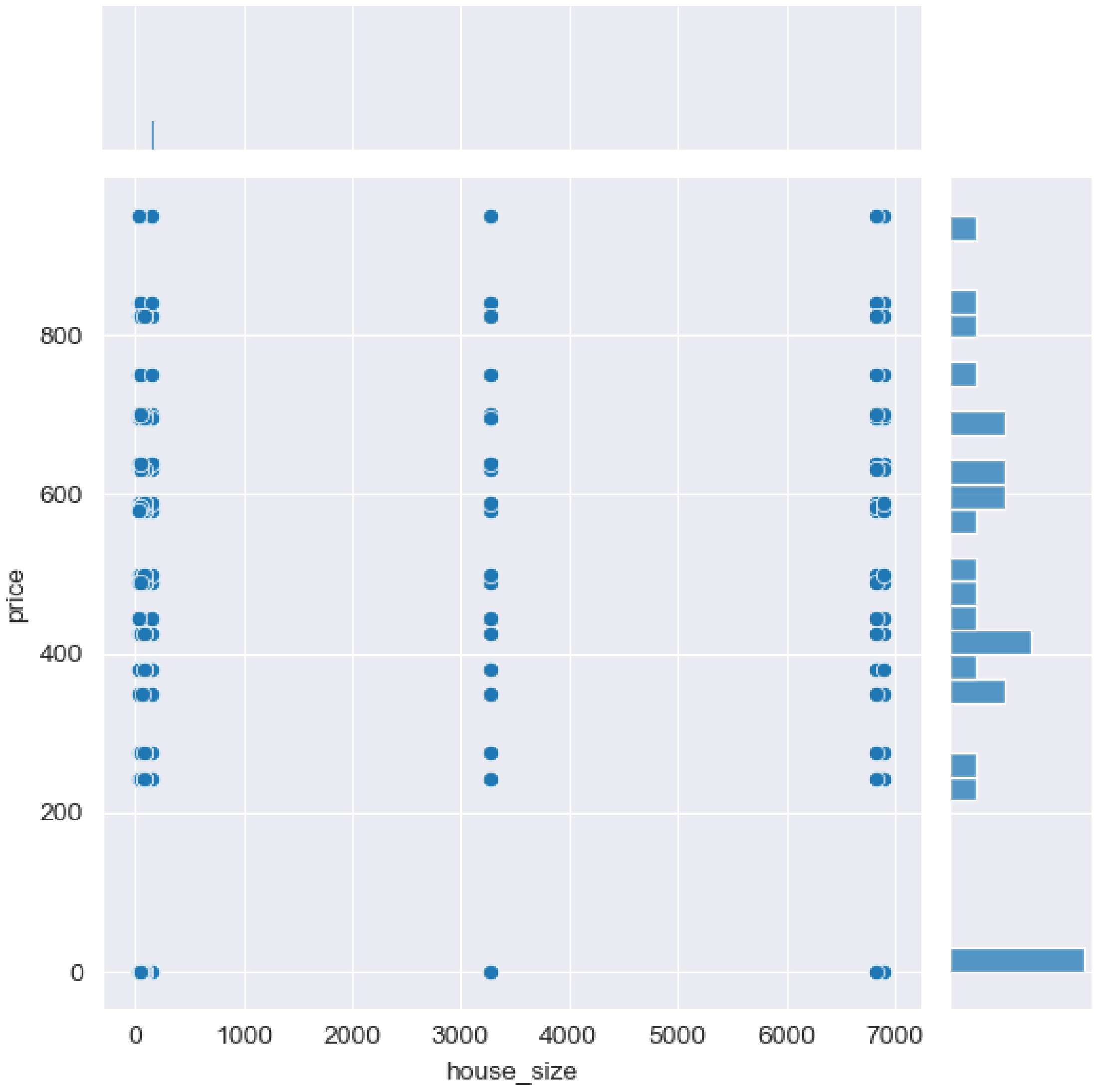


```
[37]: df_final[df_final.price<200].price.value_counts()
```

```
[37]: 1    2293
      Name: price, dtype: int64
```

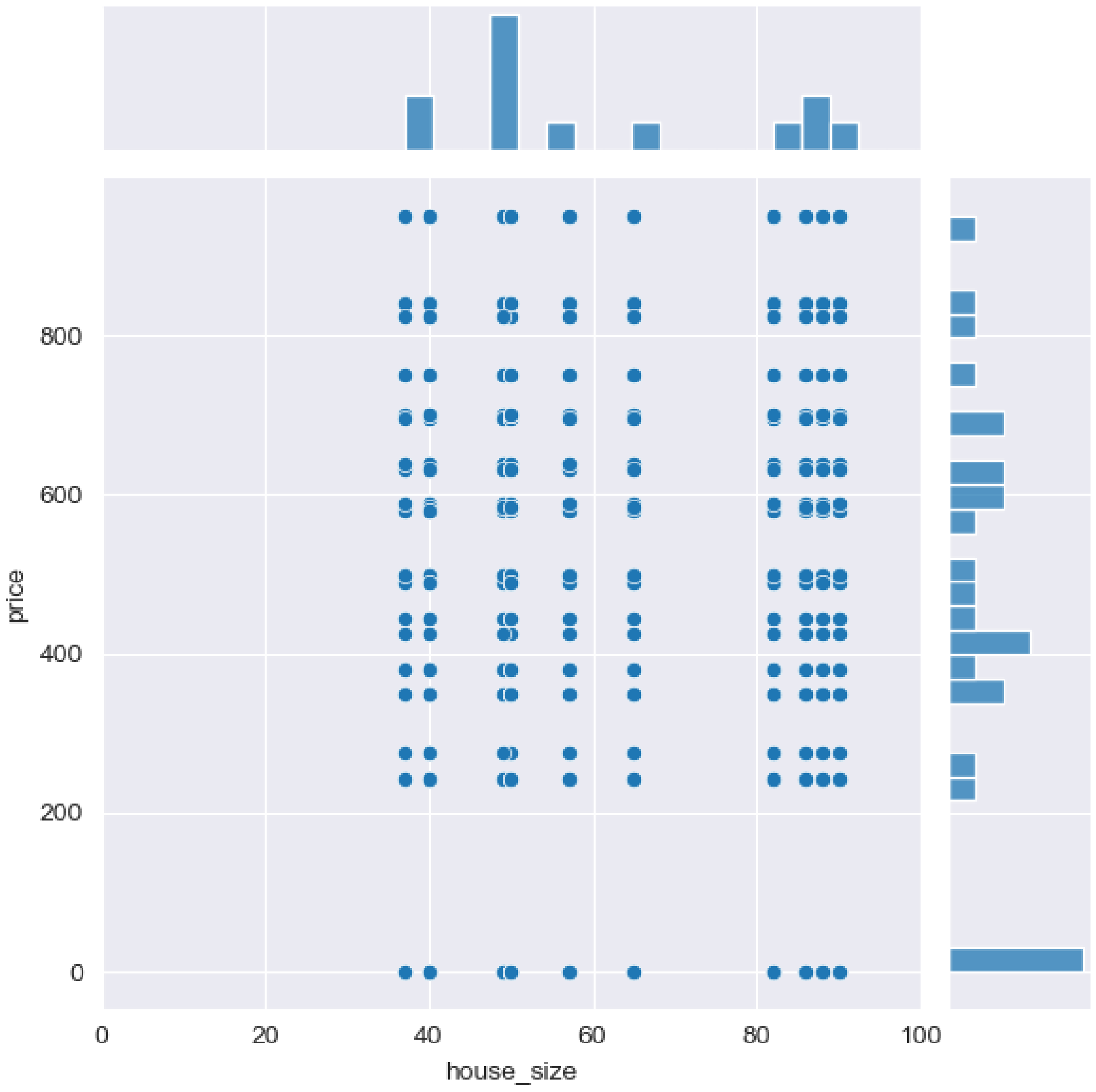
```
[38]: sns.jointplot(x='house_size',y='price',data=df_final)
```

```
[38]: <seaborn.axisgrid.JointGrid at 0x7f8c4825b7f0>
```



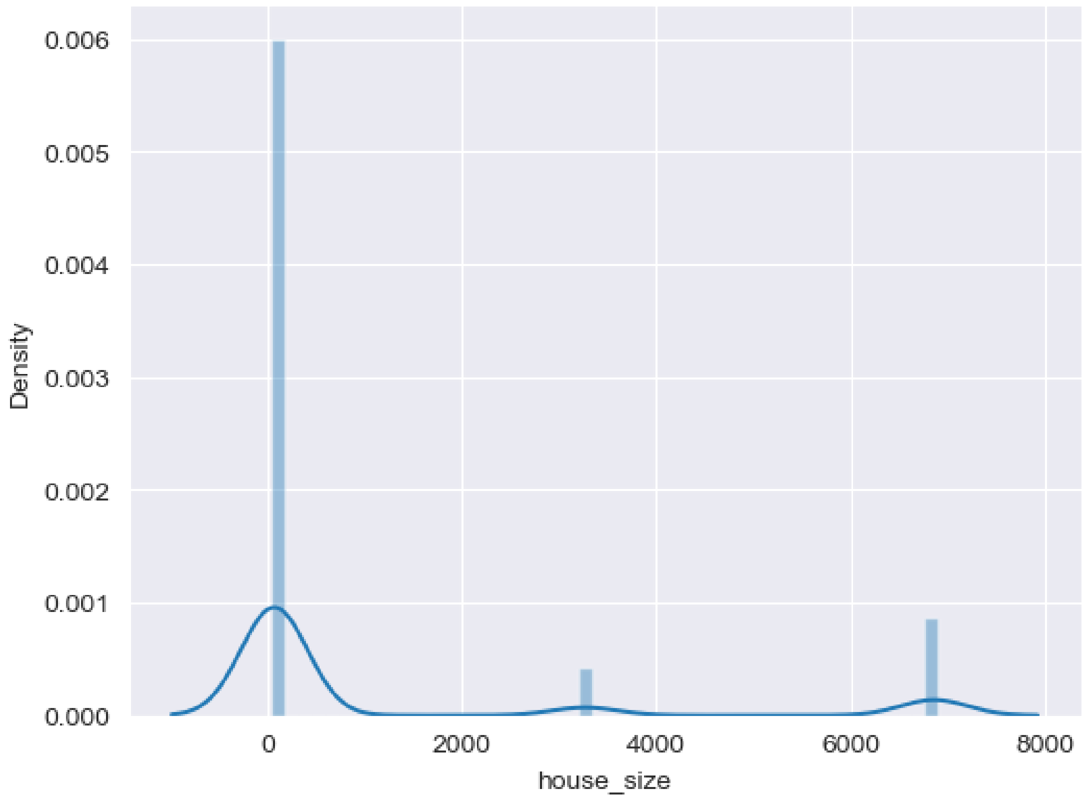
```
[39]: sns.jointplot(x='house_size',y='price',data=df_final,xlim=(0,100))
```

```
[39]: <seaborn.axisgrid.JointGrid at 0x7f8c4cb06ec0>
```



[40]:
```
sns.distplot(df_final.house_size)
```

[40]:
```
<AxesSubplot: xlabel='house_size', ylabel='Density'>
```



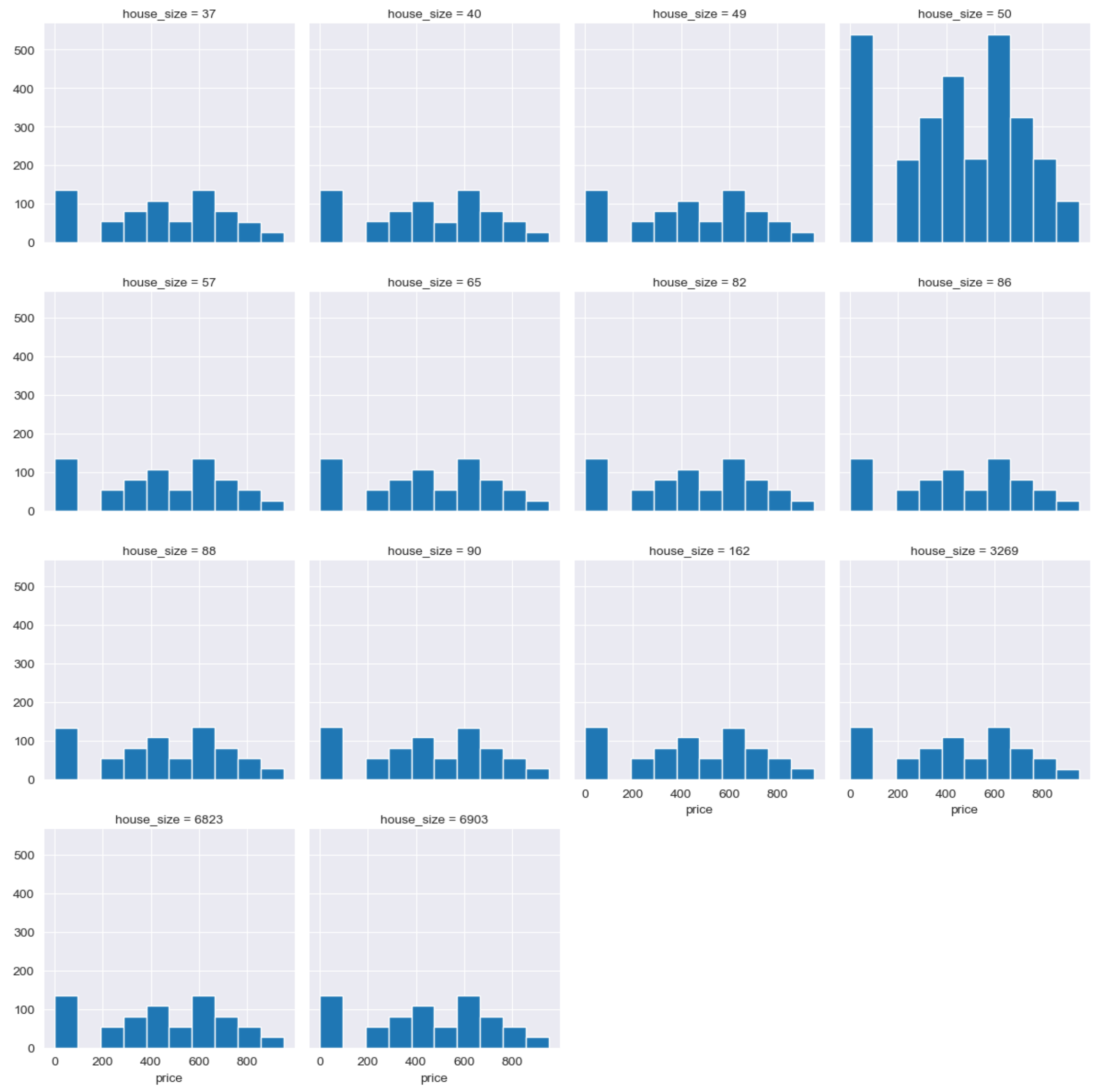
[41]:
```
df_final[df_final.house_size>80].house_size.value_counts()
```

[41]:
```
6903    729
88      728
82      728
162     728
3269    728
86      728
6823    728
90      728
Name: house_size, dtype: int64
```

[42]:
```
plt.figure(figsize=(15, 12))
g = sns.FacetGrid(df_final, col='house_size', col_wrap=4)
g.map(plt.hist, 'price')
```

[42]:
```
<seaborn.axisgrid.FacetGrid at 0x7f8c32ed14e0>
```

```
<Figure size 1500x1200 with 0 Axes>
```



7.0.1 Price Distribution by House Size

This visualization is important for several reasons

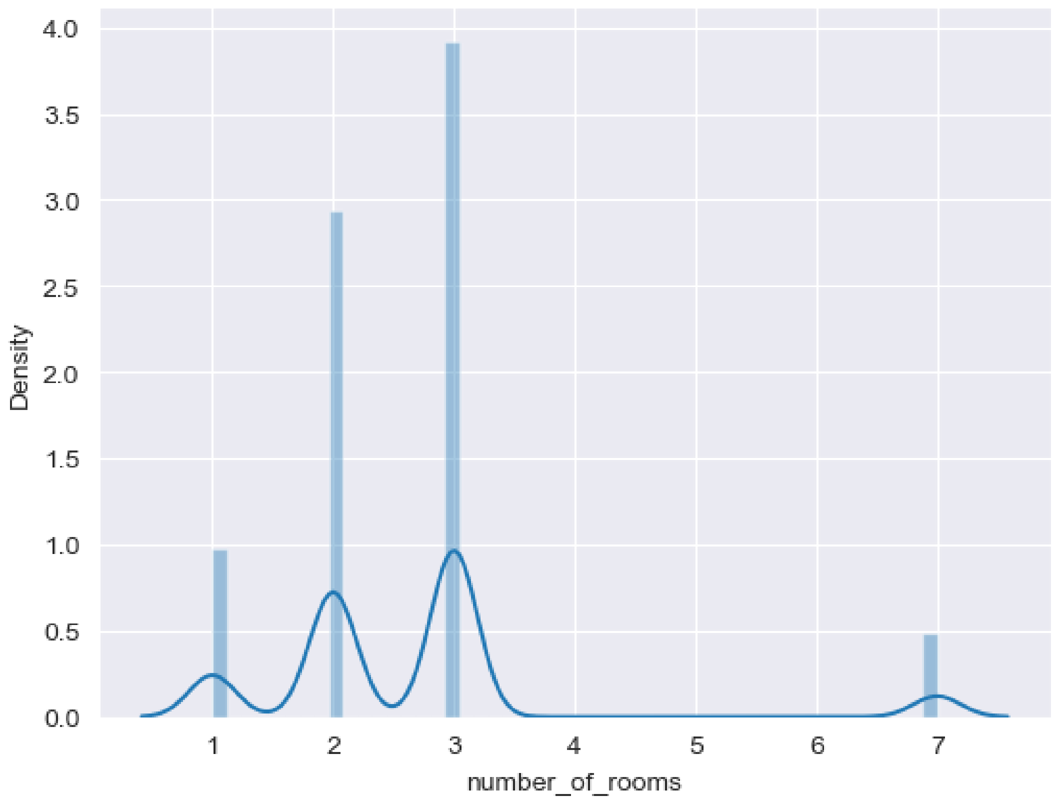
House Size Impact By examining the distribution of rental prices within different house size categories, we can gain insights into the impact of house size on rental prices. It allows us to observe patterns and variations in pricing based on the available space, helping us understand how different house sizes are priced in the market. **Identifying Price Ranges** The histogram plots within each subplot represent the distribution of prices for a particular house size category. This provides a visual representation of the price ranges within each category, allowing us to identify common price intervals or price clusters associated with different house sizes.

Market Segmentation The facet grid plot allows for market segmentation analysis. By grouping rental properties based on house size, we can identify distinct segments within the market and observe any differences in pricing patterns. This segmentation analysis can be useful for understanding market dynamics and making informed decisions regarding rental prices and housing options.

Data Validation The visualization helps validate the dataset by visually inspecting the distribution of prices within each house size category. It allows us to identify any outliers or unusual pricing patterns that may require further investigation or data cleaning steps.In summary, the facet grid plot provides a comprehensive view of the distribution of rental prices across different house size categories. This visualization helps understand the impact of house size on pricing, identify price ranges within each category, conduct market segmentation analysis, and validate the dataset. It serves as a valuable tool for gaining insights into the relationship between house size and rental prices, ultimately assisting users in making informed decisions about their housing options.

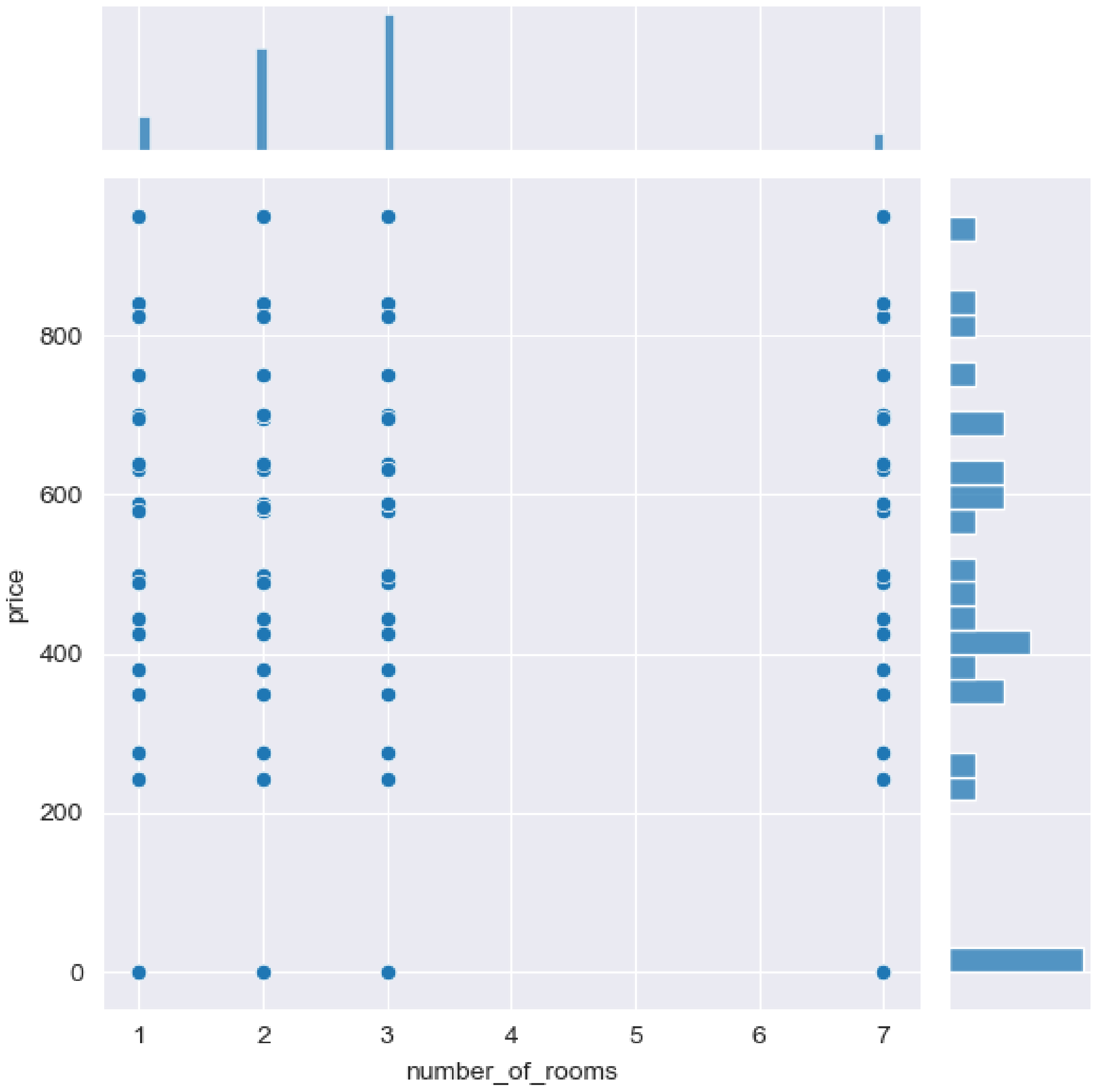
```
[43]: sns.distplot(df_final.number_of_rooms)
```

```
[43]: <AxesSubplot: xlabel='number_of_rooms', ylabel='Density'>
```



```
[44]: sns.jointplot(x='number_of_rooms',y='price',data=df_final)
```

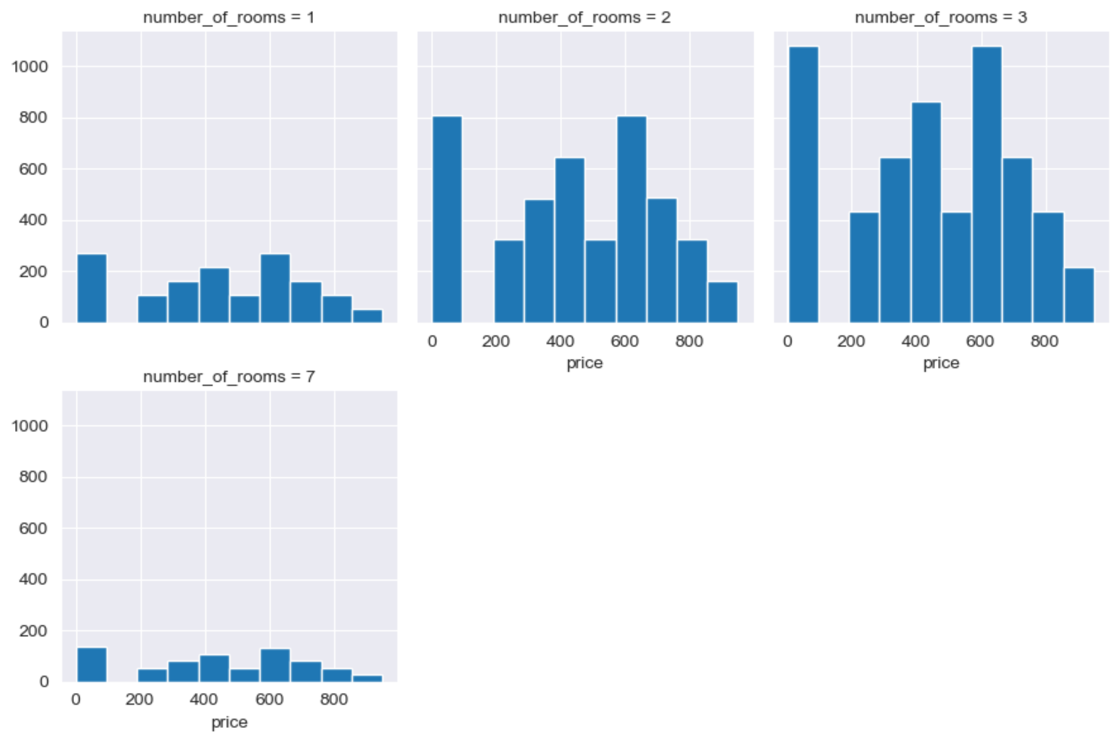
```
[44]: <seaborn.axisgrid.JointGrid at 0x7f8c35381ed0>
```



[45]:
```
plt.figure(figsize=(15, 12))
g = sns.FacetGrid(df_final, col='number_of_rooms', col_wrap=3)
g.map(plt.hist, 'price')
```

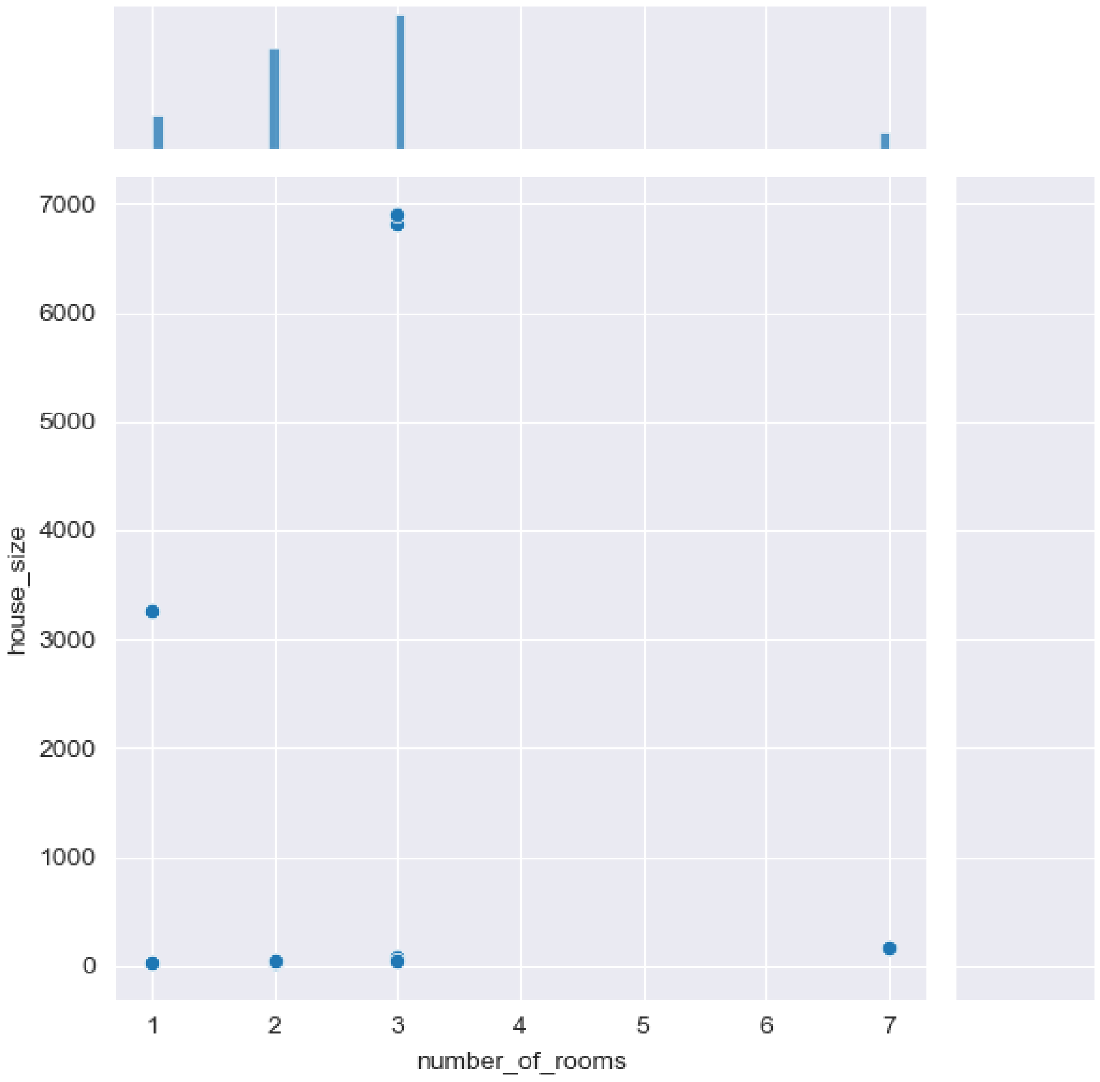
[45]:
```
<seaborn.axisgrid.FacetGrid at 0x7f8c35db2ad0>
```

```
<Figure size 1500x1200 with 0 Axes>
```



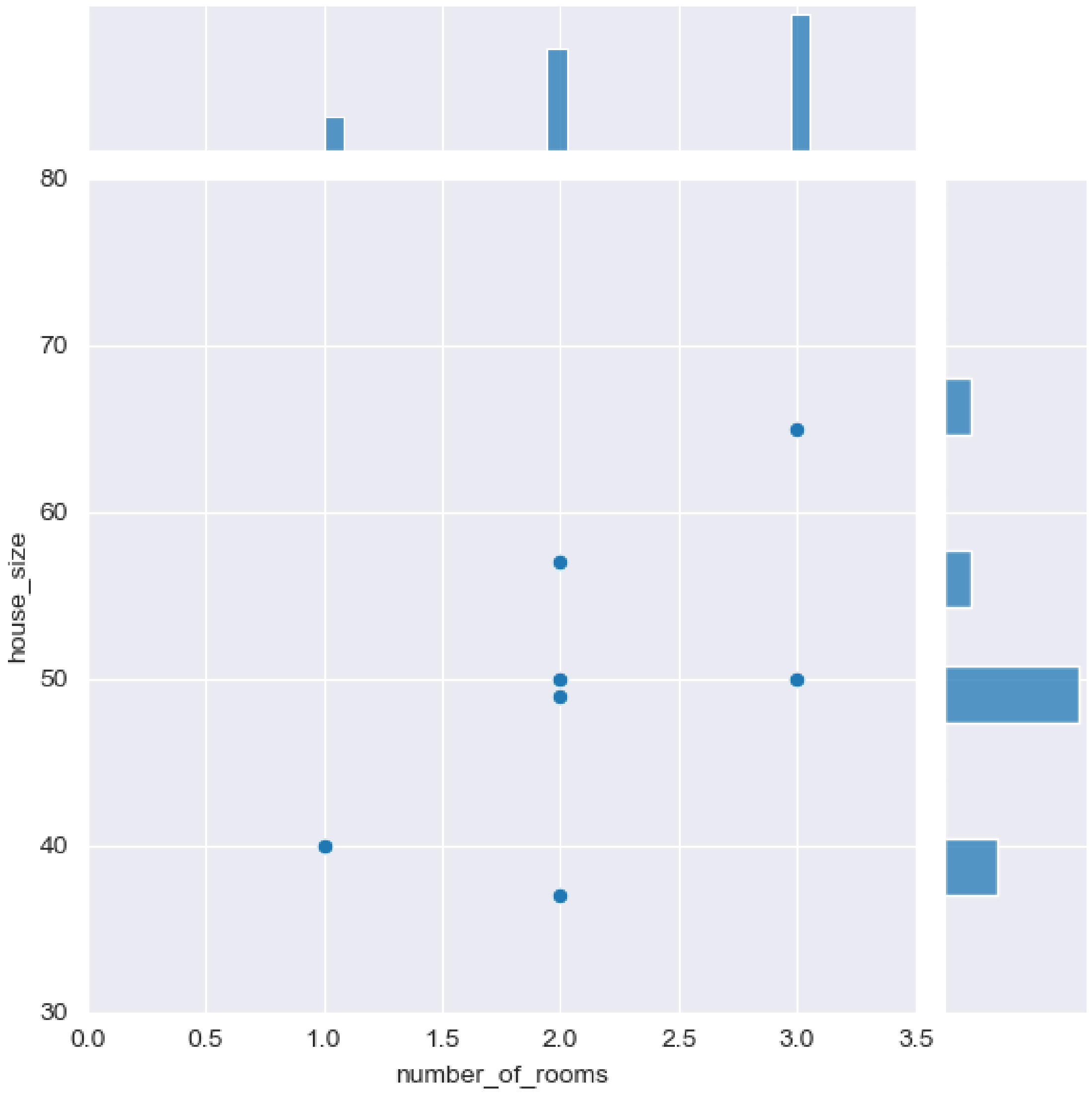
```
[46]: sns.jointplot(x='number_of_rooms',y='house_size',data=df_final)
```

```
[46]: <seaborn.axisgrid.JointGrid at 0x7f8c35e32710>
```



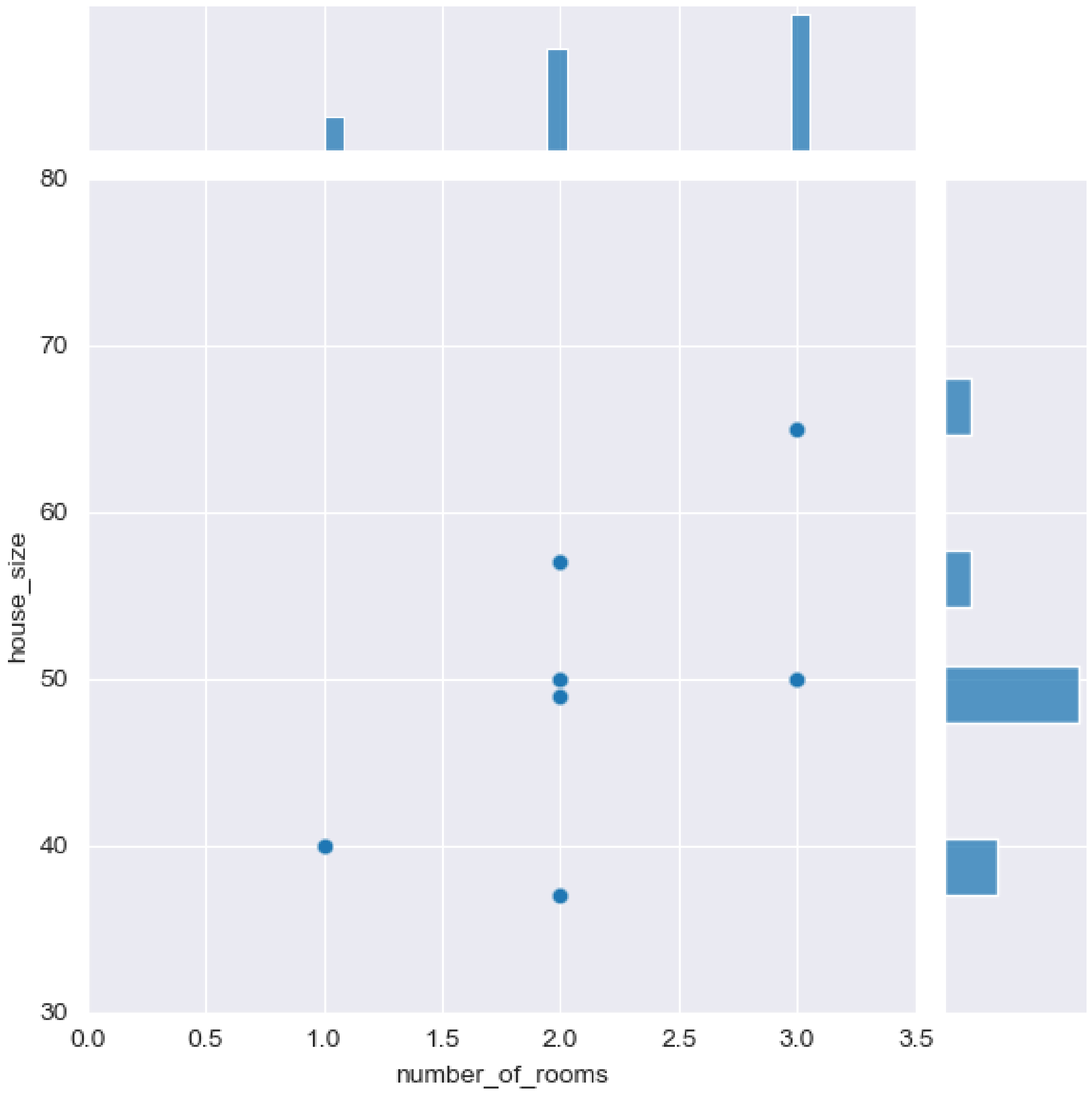
```
[47]: sns.jointplot(x='number_of_rooms',y='house_size',data=df_final,xlim=(0,3.
       ↪5),ylim=(30,80),)
```

```
[47]: <seaborn.axisgrid.JointGrid at 0x7f8c37c1a650>
```



```
[48]: sns.jointplot(x='number_of_rooms', y='house_size', data=df_final, xlim=(0, 3.
      ↪5), ylim=(30, 80), alpha=0.1)
```

```
[48]: <seaborn.axisgrid.JointGrid at 0x7f8c37eff4c0>
```



- By setting alpha=0.5, the data points in the joint plot will be more transparent, providing improved visibility of overlapping points and allowing for a clearer representation of the relationship between the 'number_of_rooms' and 'house_size' variables.Adjusting the transparency of the points can help enhance the interpretability of the joint plot, allowing for a better understanding of the density and distribution of data points in high-density areas.

[49]:
```
df_final[df_final.number_of_rooms<3.5].number_of_rooms.value_counts()
```

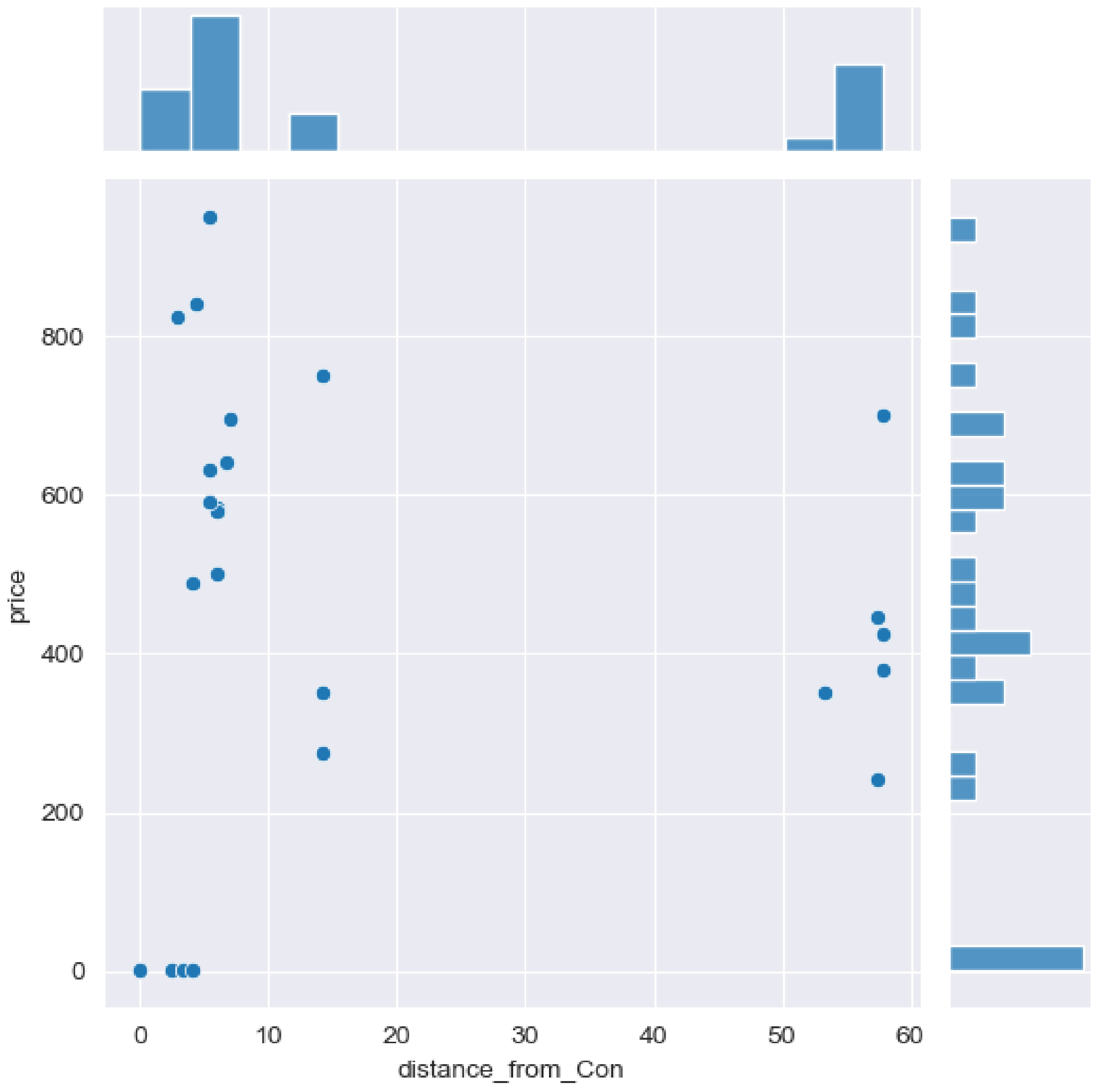
[49]:
```
3    5825
2    4369
1    1456
Name: number_of_rooms, dtype: int64
```

[50]:
```
df_final[df_final.number_of_rooms==3]
```

[50]:
```
       price                                        description  Zipcodes  \
1        242                             schöne kleine 2ZKB im DG     27580
4        580  Tolle Maisonette-Wohnung mit 2 Balkonen und Ti…       28259
8        425                         Innenstadtnahe-2-Zimmerwohnung     27568
10         1  Großzügige Hochparterre 7 Zimmer Wohnung / Bür…       28203
13       350   1-Zimmer-Apartment mit Flair und Blick ins Grüne     28717
…        …                                                  …        …
12372    823                          Zwei Zimmer in Findorffer Tor     28215
12373    425                     lichtdurchflutete-2-Zimmerwohnung     27568
12374    425                         Innenstadtnahe-2-Zimmerwohnung     27568
12375      1                   3 Zimmer Senioren Wohnung an der Weser     28217
12377    590  Schöne 3 Zimmerwohnung im Herzen von Schönebec…       28757

                   districts  house_size  number_of_rooms  text length  \
1                Bremerhaven        6903                3           24
4                  Sodenmatt          88                3           70
8                Bremerhaven          82                3           30
10                  Steintor          65                3           60
13                     Lesum          86                3           48
…                         …         …              …          …
12372  Findorff-Bürgerweide          86                3          29
12373            Bremerhaven        6823                3           33
12374            Bremerhaven          90                3           30
12375          Handelshäfen          50                3           38
12377              Vegesack        6903                3           70

       distance_from_Con
1              57.466973
4               6.042668
8              57.915749
10              0.000000
13             14.201927
…                      …
12372           2.893340
12373          57.915749
12374          57.915749
12375           4.117211
12377           5.424009

[5825 rows x 8 columns]
```

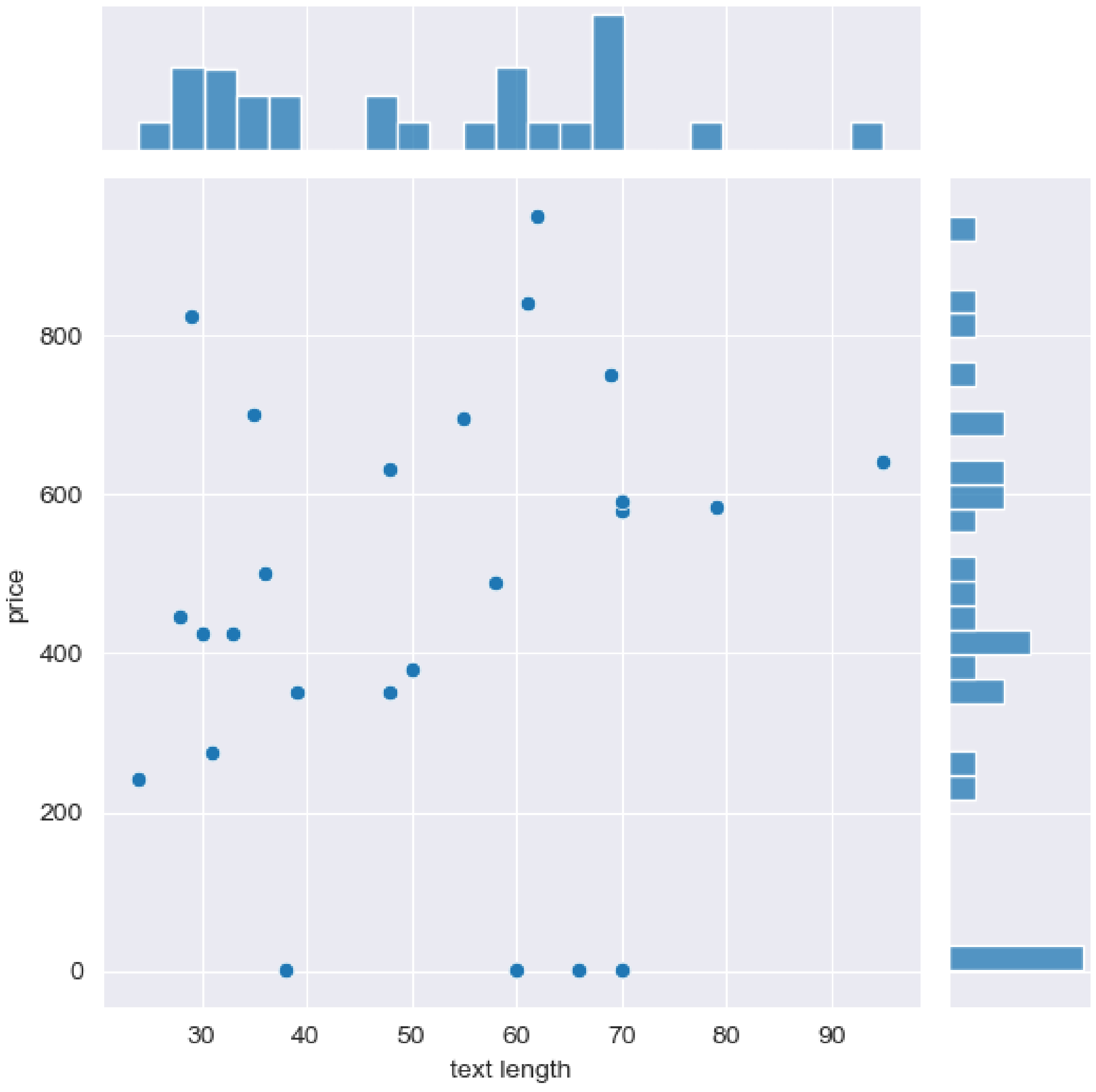
[51]:
```
sns.jointplot(x='distance_from_Con',y='price',data=df_final)
```

[51]:
```
<seaborn.axisgrid.JointGrid at 0x7f8c3655f700>
```



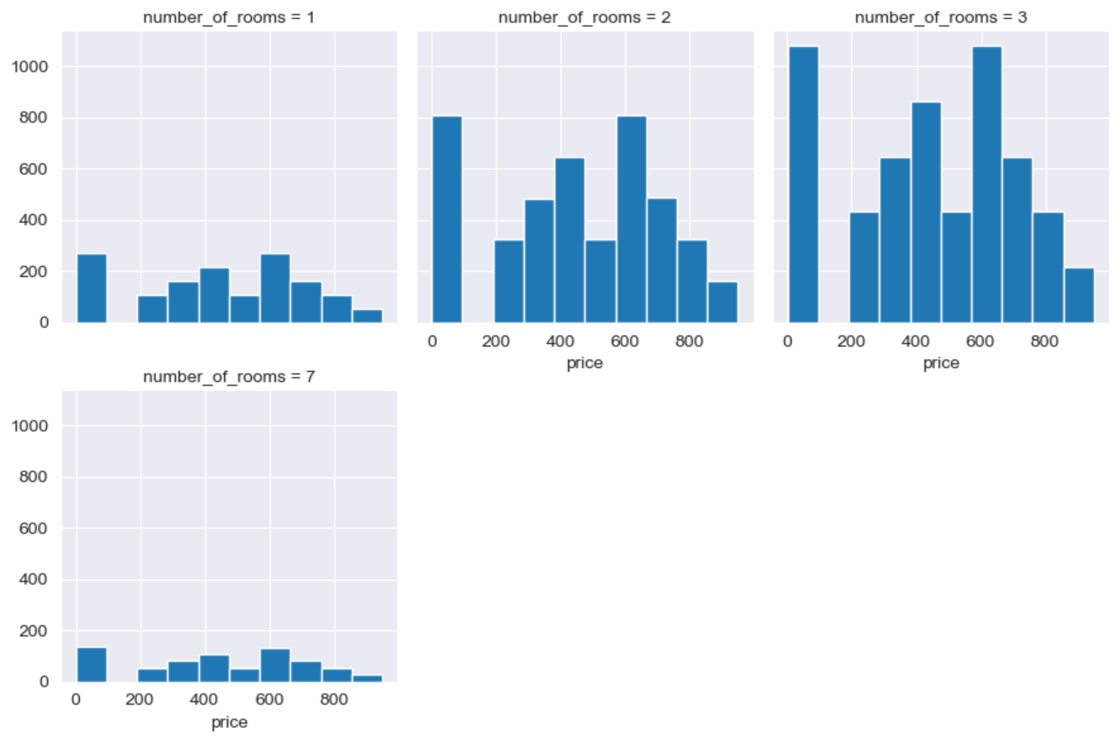
```
[52]: sns.jointplot(x='text length',y='price',data=df_final)
```

```
[52]: <seaborn.axisgrid.JointGrid at 0x7f8c3a627130>
```



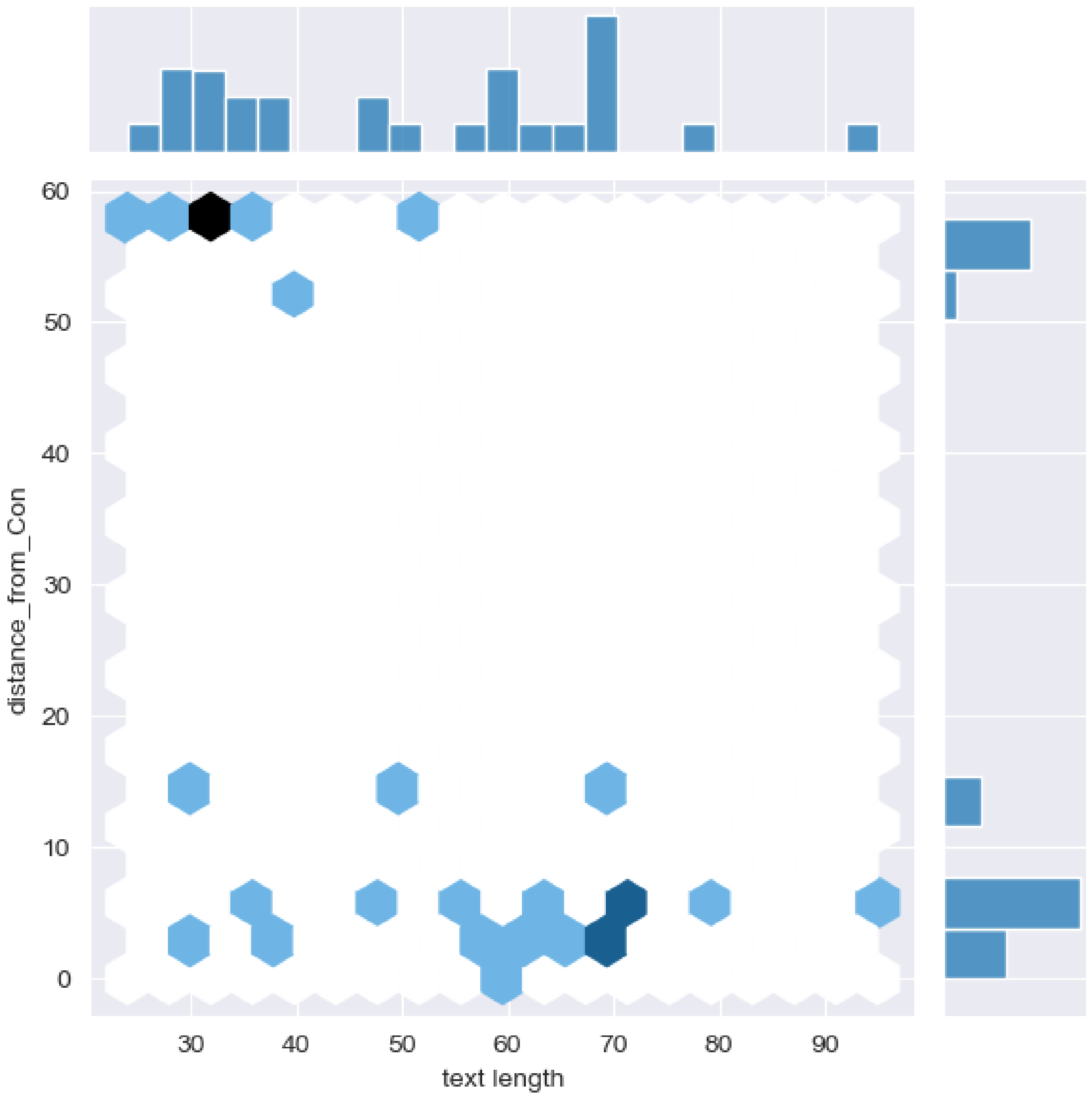
```
[53]: plt.figure(figsize=(15, 12))
      g = sns.FacetGrid(df_final, col='number_of_rooms', col_wrap=3)
      g.map(plt.hist, 'price')
```

```
[53]: <seaborn.axisgrid.FacetGrid at 0x7f8c3a943e20>

     <Figure size 1500x1200 with 0 Axes>
```



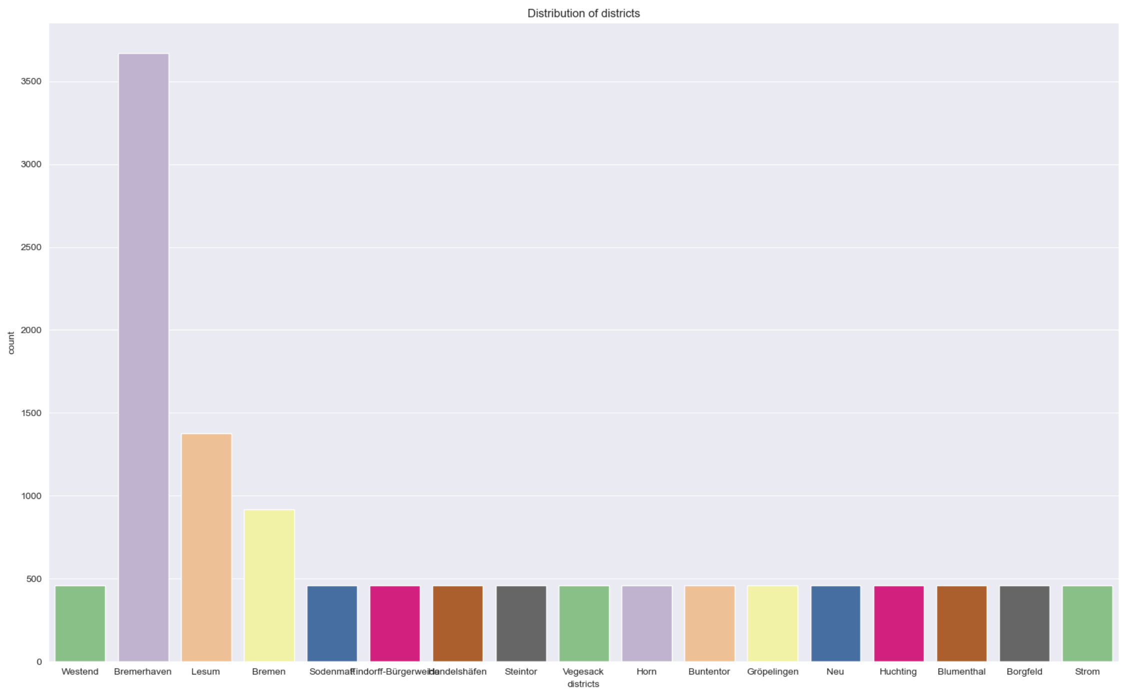
```
[54]: sns.jointplot(x='text length',y='distance_from_Con',kind='hex',data=df_final)
```

```
[54]: <seaborn.axisgrid.JointGrid at 0x7f8c3b0ea8f0>
```



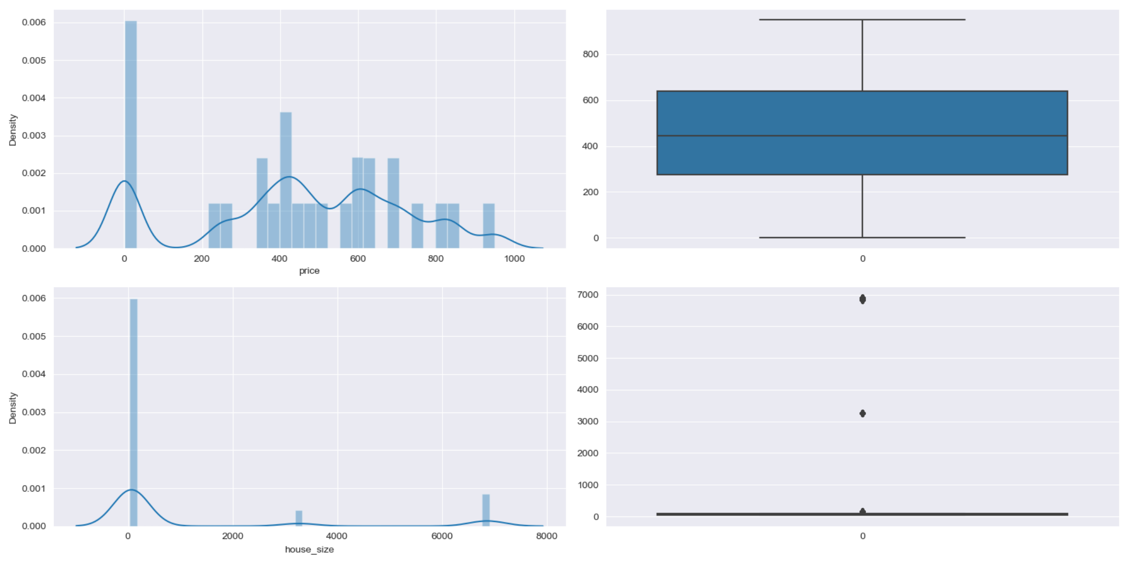
```
[55]: plt.figure(figsize=(20, 12))
      sns.countplot(x=df_final['districts'], palette='Accent')
      plt.title('Distribution of districts')
```

```
[55]: Text(0.5, 1.0, 'Distribution of districts')
```



- generates a countplot that displays the distribution of rental properties across different districts. Each bar in the plot represents the number of rental properties available in a specific district.By examining the distribution of rental properties across districts, we gain insights into the availability of housing options in different areas. It allows us to understand the geographical spread of rental properties and identify areas with a higher concentration of listings.The countplot helps identify the popularity of different districts among property owners or renters. By observing the heights of the bars, we can determine which districts have a higher number of rental properties, indicating their desirability or demand in the market.

[56]:
```
plt.figure(figsize=(16,8))

plt.subplot(2,2,1)
sns.distplot(df['price'])
plt.subplot(2,2,2)
sns.boxplot(df['price'])
plt.subplot(2,2,3)
sns.distplot(df['house_size'])
plt.subplot(2,2,4)
sns.boxplot(df['house_size'])
plt.tight_layout()
```



7.0.2 Price and House Size Distribution

The code snippet provided generates a figure with four subplots to visualize the distribution and spread of the 'price' and 'house_size' variables in the dataset.

Subplot 1 Price Distribution In the first subplot, a distribution plot (histogram) is displayed to visualize the distribution of rental prices ('price' variable). This plot provides insights into the overall shape of the price distribution, including any skewness or multimodality.

Subplot 2 Price Boxplot The second subplot presents a boxplot of the rental prices. The boxplot allows for a visual representation of the distribution's central tendency (median) and spread (interquartile range). It also helps identify potential outliers in the price variable.

Subplot 3 House Size Distribution The third subplot displays a distribution plot of the house sizes ('house_size' variable). This plot illustrates the distribution of property sizes and provides insights into the range of house sizes available in the dataset.

Subplot 4 House Size Boxplot The fourth subplot presents a boxplot of the house sizes. Similar to the previous subplot, this plot depicts the central tendency (median) and spread (interquartile range) of the house size distribution. It can help identify any potential outliers in the house size variable.By combining these four subplots, we can gain a better understanding of the distribution, spread, and potential outliers in both the rental prices and house sizes. This visualization aids in identifying any anomalies, assessing the overall distribution shape, and gaining insights into the range of prices and sizes available in the dataset.

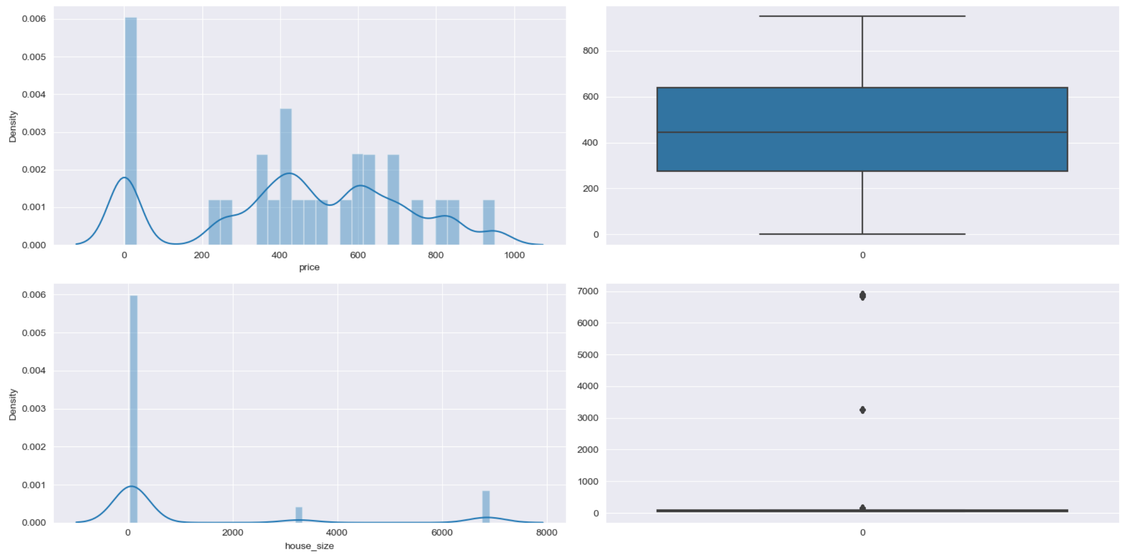
```
[57]: df_final[df_final.house_size>1000].house_size.value_counts()
```

```
[57]: 6903    729
      3269    728
      6823    728
      Name: house_size, dtype: int64
```

[58]:
```
df_final = df_final[df_final['house_size'] <= 1000]
```

- Great! By applying the code **df_final = df_final[df_final['house_size'] <= 1000]** you successfully dropped the outlier rows where the 'house_size' exceeded 1000 in the DataFrame 'df_final'. This step helps ensure that the dataset contains more accurate and representative values for house sizes, eliminating any extreme or erroneous data points that may affect the analysis or modeling process.
- Removing outliers is a crucial data preprocessing step as it allows for a more reliable and meaningful analysis. Outliers, in this case, the house sizes greater than 1000, could potentially introduce biases or distort the distribution of house sizes, leading to misleading insights or predictions. By excluding these outliers, you have enhanced the dataset's integrity, improving the overall quality of the analysis and modeling tasks performed on the SmartRentCalculator project.

[59]:
```
plt.figure(figsize=(16,8))

plt.subplot(2,2,1)
sns.distplot(df['price'])
plt.subplot(2,2,2)
sns.boxplot(df['price'])
plt.subplot(2,2,3)
sns.distplot(df['house_size'])
plt.subplot(2,2,4)
sns.boxplot(df['house_size'])
plt.tight_layout()
```



[60]:
```
df_final.head()
```

```
[60]:    price                                         description  Zipcodes  \
      0    490  Dachgeschosswohnung in Walle zu vermieten (Sin…     28217
      2    750  St. Magnus / Großzügige 3-Zimmer-Wohnung mit L…     28717
      3      1  Wohnen im Schnoor in einer Maisonette-Wohnung …     28195
      4    580  Tolle Maisonette-Wohnung mit 2 Balkonen und Ti…     28259
      5    445                          schöne große 3ZKB inkl.Küche     27580

           districts  house_size  number_of_rooms  text length  distance_from_Con
      0      Westend          50                2           58           4.117211
      2        Lesum          37                2           69          14.201927
      3       Bremen          49                2           70           3.384377
      4    Sodenmatt          88                3           70           6.042668
      5  Bremerhaven          57                2           28          57.466973
```

```
[61]: df_final.Zipcodes.nunique()
```

```
[61]: 17
```

- The presence of 17 unique zip codes suggests that the dataset covers a range of geographic areas, potentially encompassing different neighborhoods or districts within a specific location. This level of diversity in zip codes provides users of the SmartRentCalculator tool with a variety of options when searching for rental properties, allowing them to consider different locations based on their preferences and needs.
- By having rental properties from multiple zip codes, the dataset becomes more representative of the overall rental market and offers a comprehensive view of housing options in various areas. This information can assist users in narrowing down their search and finding suitable rental properties in specific zip code regions that align with their desired locations or amenities.

```
[ ]:
```

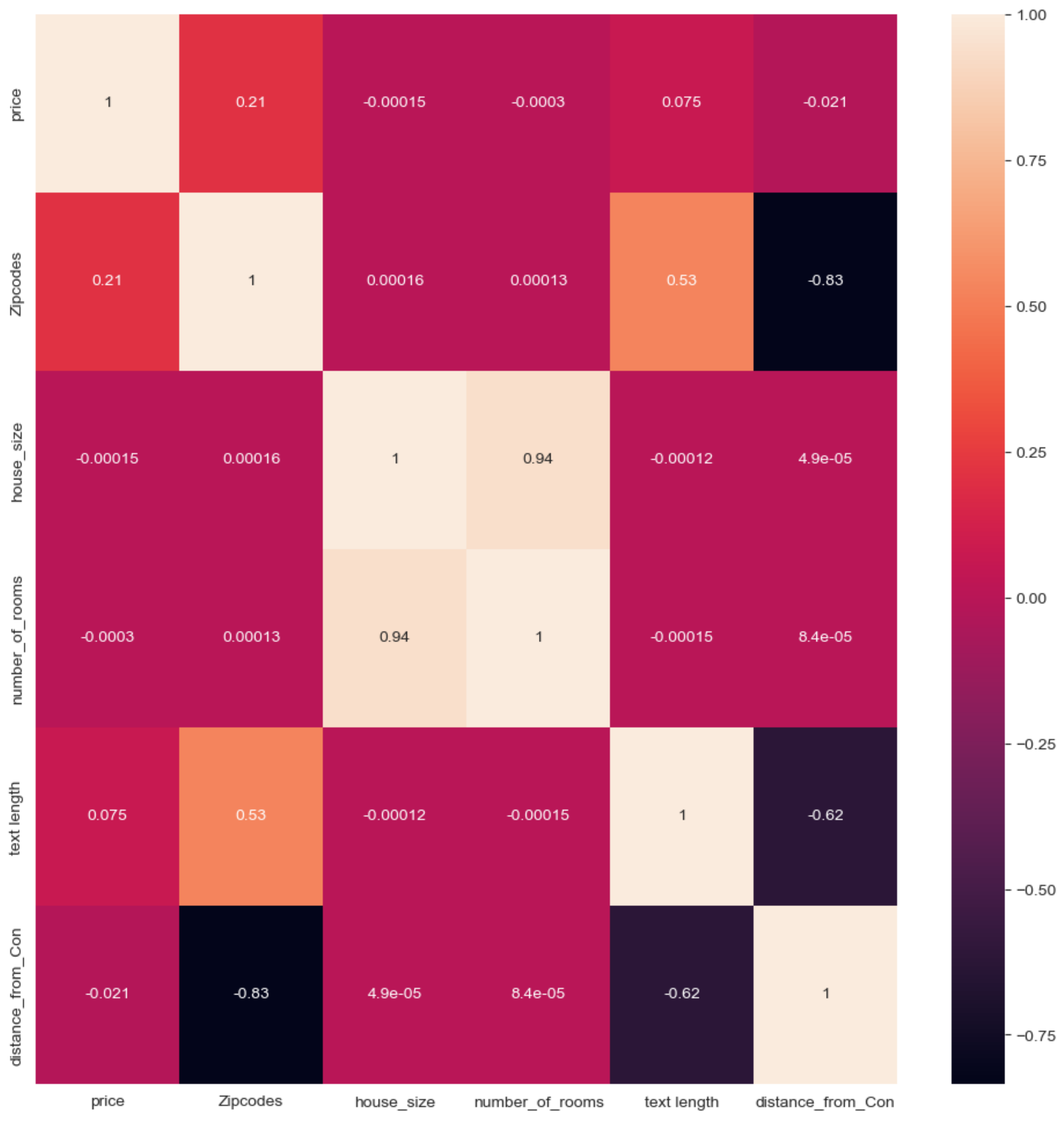
```
[62]: #dict_zip={}

      #for zipcode in Zipcodes_list:
          #my_city_county = zcode.matching(zipcode.astype('str'))
          #if len(my_city_county)==1: # if  zipcode is present then get county else,␣
       ↪assign zipcode to county
              #county=my_city_county[0].get('county')
          #else:
              #county=zipcode

          #dict_zip.update({zipcode:county})
```

```
[63]: df_final
```

```
[63]:        price                                        description  Zipcodes  \
      0        490  Dachgeschosswohnung in Walle zu vermieten (Sin…     28217
      2        750  St. Magnus / Großzügige 3-Zimmer-Wohnung mit L…     28717
      3          1  Wohnen im Schnoor in einer Maisonette-Wohnung …     28195
      4        580  Tolle Maisonette-Wohnung mit 2 Balkonen und Ti…     28259
      5        445                       schöne große 3ZKB inkl.Küche     27580
      …        …                                                …       …
      12370    580  Tolle Maisonette-Wohnung mit 2 Balkonen und Ti…     28259
      12372    823                         Zwei Zimmer in Findorffer Tor     28215
      12374    425                        Innenstadtnahe-2-Zimmerwohnung     27568
      12375      1                3 Zimmer Senioren Wohnung an der Weser     28217
      12376      1  Großzügige Hochparterre 7 Zimmer Wohnung / Bür…     28203

                        districts  house_size  number_of_rooms  text length  \
      0                   Westend          50                2           58
      2                     Lesum          37                2           69
      3                    Bremen          49                2           70
      4                 Sodenmatt          88                3           70
      5               Bremerhaven          57                2           28
      …                       …         …                …          …
      12370             Sodenmatt          40                1           70
      12372  Findorff-Bürgerweide        86                3          29
      12374           Bremerhaven          90                3           30
      12375          Handelshäfen          50                3           38
      12376              Steintor          50                2           60

             distance_from_Con
      0               4.117211
      2              14.201927
      3               3.384377
      4               6.042668
      5              57.466973
      …                    …
      12370           6.042668
      12372           2.893340
      12374          57.915749
      12375           4.117211
      12376           0.000000

      [10193 rows x 8 columns]
```

```
[64]: plt.figure(figsize=(12,12))
      sns.heatmap(df_final.corr(),annot=True)
      plt.tight_layout
```

```
[64]: <function matplotlib.pyplot.tight_layout(*, pad=1.08, h_pad=None, w_pad=None,
      rect=None)>
```

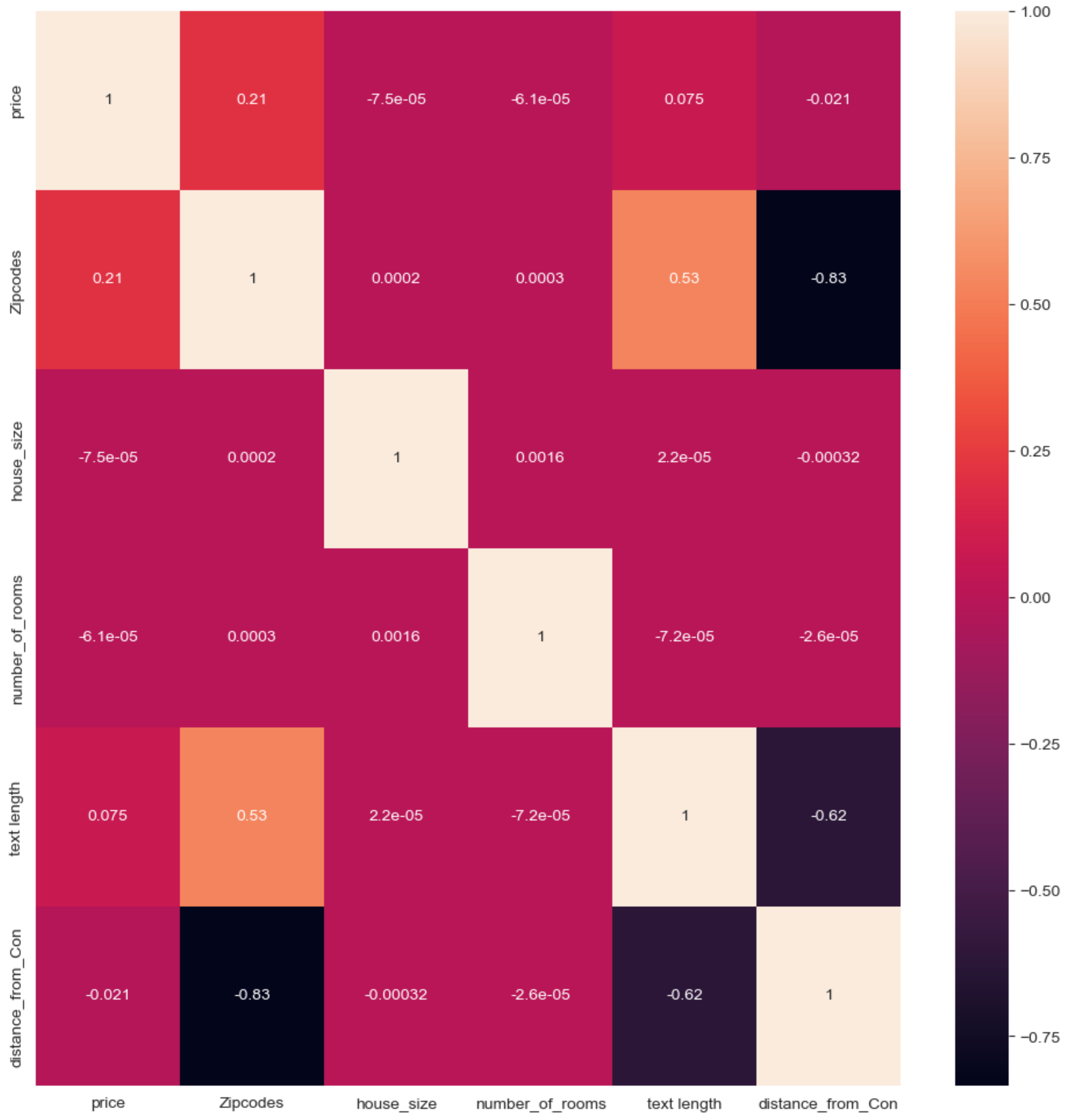


[65]:
```
df_final.columns
```

[65]:
```
Index(['price', 'description', 'Zipcodes', 'districts', 'house_size',
       'number_of_rooms', 'text length', 'distance_from_Con'],
      dtype='object')
```

[66]:
```
plt.figure(figsize=(12,12))
sns.heatmap(df.corr(),annot=True)
plt.tight_layout
```

```
[66]: <function matplotlib.pyplot.tight_layout(*, pad=1.08, h_pad=None, w_pad=None,
      rect=None)>
```



8 8 Data Analysis

```
[67]: df_final.corr()['price'].sort_values()
```

```
[67]: distance_from_Con   -0.021437
      number_of_rooms     -0.000298
      house_size          -0.000147
```

```
text length          0.074847
Zipcodes             0.206514
price                1.000000
Name: price, dtype: float64
```

8.1 8.1 Feature Selection and Correlation Analysis

In the SmartRentCalculator project, feature selection played a crucial role in developing an effective predictive model for rental prices. By analyzing the correlation between the target variable (price) and other features, we gained insights into the relationships and importance of each feature in predicting rental prices. Here are the key findings from the correlation analysis:

- **Distance from Constructor University (distance_from_Con)**: The correlation coefficient between distance_from_Con and price is approximately -0.021. This indicates a weak negative correlation, suggesting that there is a slight tendency for rental prices to decrease as the distance from the university increases. However, the correlation is not significant, indicating that distance_from_Con alone may not be a strong predictor of rental prices.
- **Number of Rooms (number_of_rooms)** and **House Size (house_size)**: The correlation coefficients for both number_of_rooms and house_size are close to zero, indicating a negligible correlation with rental prices. These features may not be strong predictors of rental prices in the current model.
- **Text Length (text length)**: The correlation coefficient between text length and price is approximately 0.075. This suggests a weak positive correlation, indicating that rental prices may slightly increase with longer property descriptions. However, the correlation is relatively low, indicating that text length alone may not have a strong impact on rental prices.
- **Zipcodes**: The correlation coefficient between Zipcodes and price is approximately 0.207. This indicates a moderate positive correlation, suggesting that there is a tendency for rental prices to increase in certain zip code areas. Zipcodes can be considered as a relevant feature in predicting rental prices.

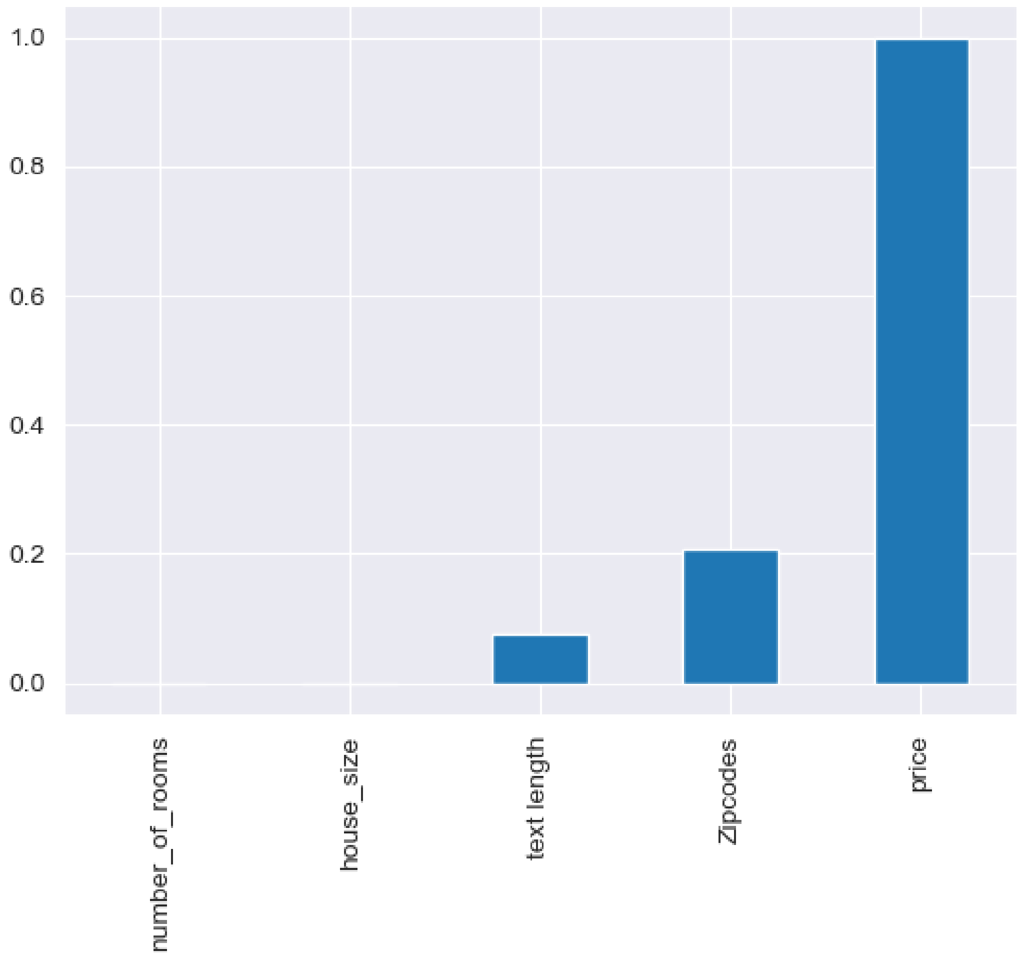
Based on the correlation analysis, it is evident that not all features have a significant impact on rental prices. The correlation coefficients reveal the strength and direction of the relationships, helping us in feature selection. Features such as distance_from_Con, number_of_rooms, and house_size have weak or negligible correlations with price, indicating that they may not contribute significantly to the predictive power of the model. On the other hand, features like text length and zipcodes show a relatively stronger correlation, suggesting their potential relevance in predicting rental prices.

During the feature selection process, we considered these correlations along with other factors such as domain knowledge, significance, and interpretability of the features. This helped us select the most relevant and influential features to include in the predictive model, improving its accuracy and performance.

Feature selection is an important step in developing a predictive model as it allows us to focus on the most significant variables and avoid including irrelevant or redundant features. By considering correlations and other relevant factors, we ensure that the selected features have a meaningful impact on the model's ability to predict rental prices accurately.

```
[68]: df_final.corr()['price'][:-1].sort_values().plot(kind='bar')
```

```
[68]: <AxesSubplot: >
```



8.2 8.2 Model Selection

```
[ ]:
```

```
[69]: import string
from nltk.corpus import stopwords

def text_process(mess):

    nopunc = [char for char in mess if char not in string.punctuation]

    nopunc = ''.join(nopunc)
    return [word for word in nopunc.split() if word.lower() not in stopwords.
↪words('German')]
```

8.2.1 8.2.1Text Processing

Text processing is a vital step in the SmartRentCalculator project, where we analyze property descriptions and extract valuable information. The function `text_process` employs several techniques to clean and preprocess the text data. Let's discuss the code and its significance in the project:

Removing Punctuation The code utilizes the `string.punctuation` module to remove punctuation marks from the property descriptions. Punctuation marks, such as periods, commas, and quotation marks, do not contribute to the content's meaning. Removing them ensures that the text is cleaner and more suitable for analysis.

Example Original text: "This spacious, fully furnished apartment is located in a quiet neighborhood, close to amenities." Processed text: "This spacious fully furnished apartment is located in a quiet neighborhood close to amenities"

Joining Characters After removing punctuation, the characters are joined back together to form a single string. This step ensures that the text is in the correct format for further processing.

Example: Original text: "This spacious fully furnished apartment is located in a quiet neighborhood close to amenities" Processed text: "This spacious fully furnished apartment is located in a quiet neighborhood close to amenities"

Removing Stopwords Stopwords are common words that do not carry significant meaning and can be safely ignored in the analysis. The code uses the `stopwords.words('German')` function to obtain a list of German stopwords and removes them from the text data. By eliminating stopwords, we focus on the more meaningful and informative words in the property descriptions.

Example: Original text: "This spacious fully furnished apartment is located in a quiet neighborhood close to amenities" Processed text: "spacious fully furnished apartment located quiet neighborhood close amenities"

Text processing is crucial in the project for the following reasons:

Feature Extraction Property descriptions often contain valuable details such as amenities, location advantages, and unique features. By processing the text, we extract these features and use them as additional predictors in the predictive model. This enhances the accuracy and relevance of rental price predictions.

Noise Reduction: Property descriptions may include irrelevant information or noise in the form of punctuation marks, symbols, or common words. By removing punctuation and stopwords, we reduce the noise and ensure that the analysis focuses on the most relevant and informative content. This improves the quality of the data used for modeling and enhances the effectiveness of the predictive model.

Enhanced Understanding: Text processing enables us to gain a better understanding of the property descriptions and extract meaningful insights. By cleaning and preprocessing the text data, we can identify patterns, keywords, and phrases that are relevant to rental prices. This understanding aids in feature engineering, model development, and generating accurate predictions.

[75]:
```python
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
```

[71]:
```python
from sklearn.ensemble import RandomForestRegressor
rfr =RandomForestRegressor()
```

[ ]:
```python

```

[72]:
```python
df = pd.DataFrame(df_final[['price', 'description','house_size',␣
↪'number_of_rooms', 'text length', 'distance_from_Con']])
```

8.2.2 Column Removal and Reasons

In the code provided, we removed some columns from the dataset (`df_final`) to create a new DataFrame (`df`) for the model. Here are the removed columns and the reasons for their removal:

'Unnamed: 0': This column appears to be an index or an identifier column that was not relevant to the analysis or modeling process. It does not provide any meaningful information about the rental prices or property features, so it was removed.**'Zipcodes'**: The column 'Zipcodes' represents the zip codes associated with each rental property. While zip codes can be valuable for certain analyses, in the context of our model, we already have the 'distance_from_Con' feature that captures the proximity to Constructor University. Considering that 'distance_from_Con' already accounts for location information, including zip codes would introduce redundancy and potentially increase the dimensionality of the dataset. Therefore, the 'Zipcodes' column was removed **'districts'**: Similar to the 'Zipcodes' column, the 'districts' column provides additional location information. However, since we already have the 'distance_from_Con' feature, which captures the proximity to Constructor University, including 'districts' would introduce redundancy. Therefore, the 'districts' column was removed. **Irrelevance**: The removed columns did not contain information that was directly relevant to predicting rental prices. For example, the 'Unnamed: 0' column appeared to be an indexing column, and the 'Zipcodes' and 'districts' columns were redundant given the presence of the 'distance_from_Con' feature. **Dimensionality Reduction**: By removing these columns, we reduced the dimensionality of the dataset. Reducing the number of features can help prevent overfitting, improve computational efficiency, and simplify the modeling process.**Redundancy**: The removed columns duplicated information already captured by other features. By eliminating redundant columns, we avoid multicollinearity issues and keep the dataset focused on the most informative and unique predictors.

[ ]:
```python

```

8.2.3 8.3 Model 1 - LinearRegression Model Training

For the SmartRentCalculator project, I trained a Linear Regression model using scikit-learn's `Pipeline` and (ColumnTransformer) classes. Here is an overview of the steps involved in training and evaluating the model:

- **Data Splitting** I split the dataset (df) into training and testing sets using the (train_test_split) function. By dropping the target variable (‘price’) from the DataFrame, I obtained the predictors (X_train, X_test), and assigned the target variable to (y_train and y_test). This ensured that I had separate datasets for training and evaluating the model.

- **Data Preprocessing** I created two pipelines using **Pipeline** objects. The **num_transformer** pipeline applied standard scaling to the numerical features (‘house_size’, ‘number_of_rooms’, ‘text length’, ‘distance_from_Con’). The **text_transformer** pipeline utilized the **CountVectorizer** with the **text_process** function from the previous section to process and transform the ‘description’ feature.
- **Column Transformation** I combined the numerical and text pipelines using the **ColumnTransformer** to create the **preprocessor**. This transformer applied the respective transformations to the specified columns (‘house_size’, ‘number_of_rooms’, ‘text length’, ‘distance_from_Con’ for numerical features and ‘description’ for text features).
- **Model Definition** I defined the model as a **Pipeline** with the **preprocessor** as the first step and a **LinearRegression** model as the second step. This pipeline ensured that the preprocessing steps were applied to the data before training the regression model.
- **Model Training and Prediction** I trained the model using the training data (X_train, y_train) by calling the **fit** method. After training, I made predictions on the testing data (X_test) using the **predict** method, and stored the predicted prices in predic.
- **Model Evaluation** I evaluated the performance of the model using the R2 score, which measures the proportion of the variance in the target variable that is predictable from the predictors. I calculated the R2 score using the **score** method of the model and printed it as the output.

By utilizing the pipeline and column transformer, I created a streamlined workflow for data preprocessing and model training. This approach allowed me to handle both numerical and text features separately and apply the necessary transformations consistently. The evaluation of the model’s performance using the R2 score provided an indication of how well the model fit the data and its ability to explain the variance in rental prices.

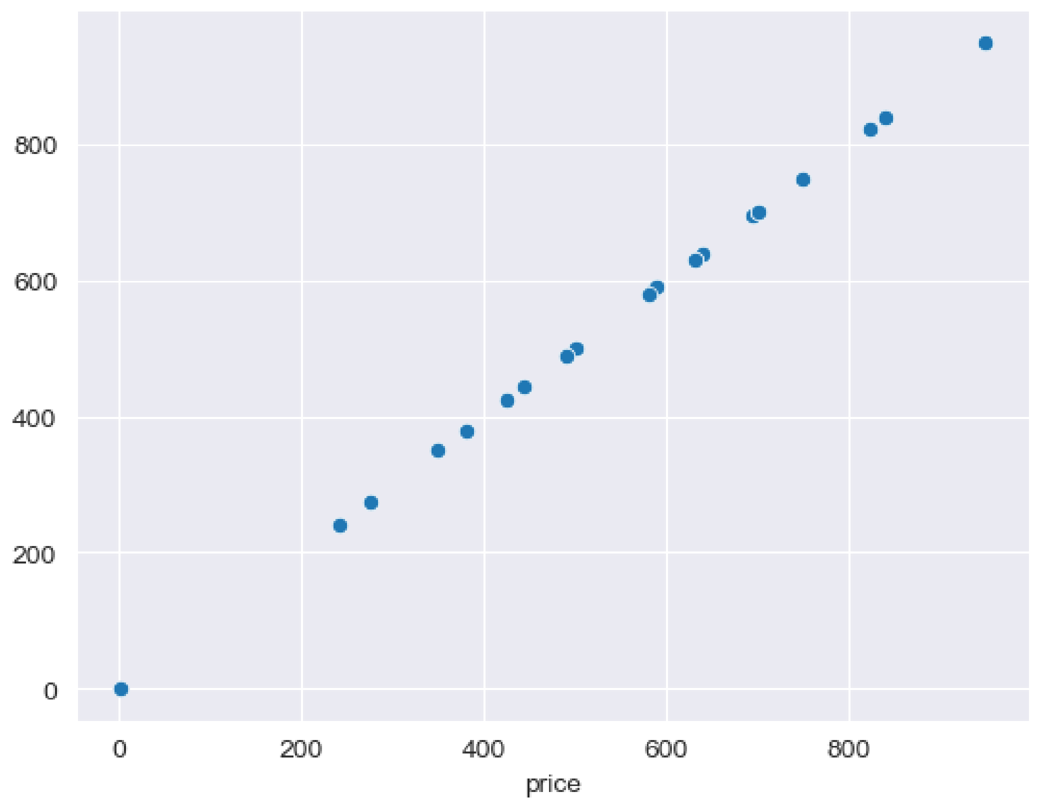
[78]:
```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer


X_train, X_test, y_train, y_test = train_test_split(df.drop('price', axis=1),␣
 ↪df['price'], test_size=0.4, random_state=42)


num_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])


text_transformer = Pipeline(steps=[
    ('vectorizer', CountVectorizer(analyzer=text_process))
])


preprocessor = ColumnTransformer(
    transformers=[
        ('num', num_transformer, ['house_size', 'number_of_rooms', 'text␣
 ↪length', 'distance_from_Con']),
        ('text', TfidfVectorizer(), 'description')
    ])


model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])


model.fit(X_train, y_train)
predic1 = model.predict(X_test)

score = model.score(X_test, y_test)
print(f"Model R2 score: {score}")
```

```
Model R2 score: 0.999999999916466
```

8.2.4 Model 1 Evaluation

[79]:
```
sns.scatterplot(x=y_test,y=predic1)
```

[79]:
```
<AxesSubplot: xlabel='price'>
```

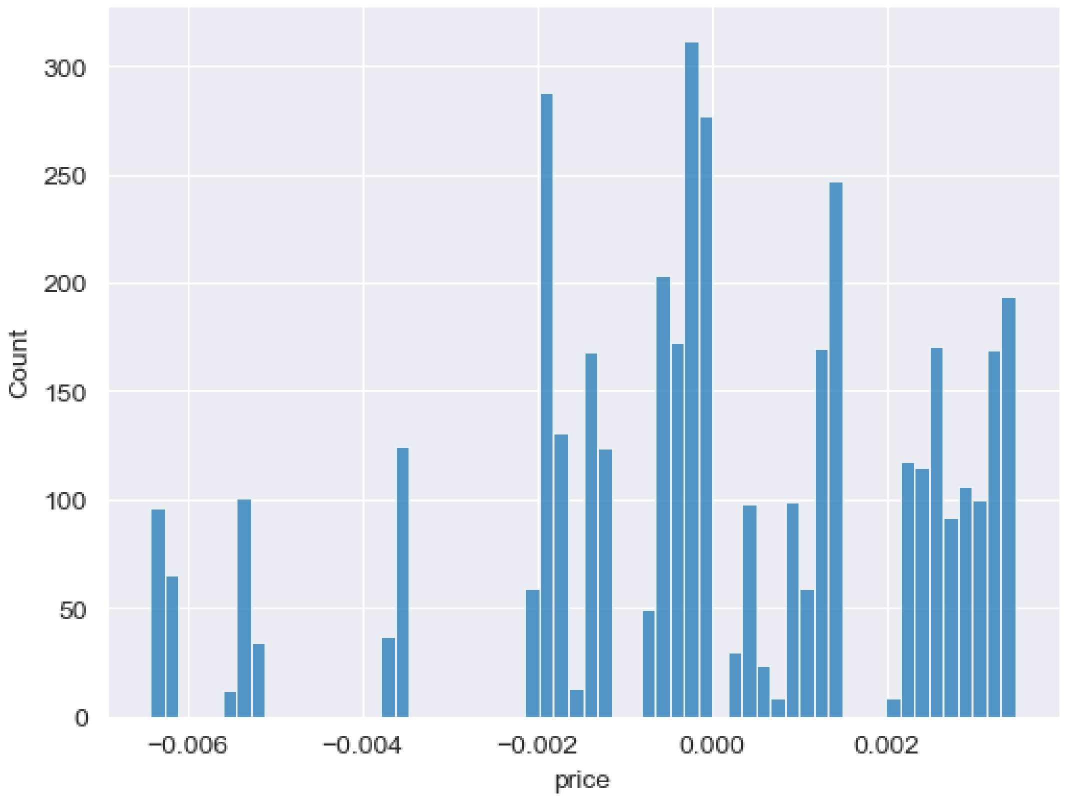


Scatterplot of Predicted vs. Actual Prices

- To evaluate the performance of Model 1, I created a scatterplot using seaborn's `scatterplot` function. The scatterplot compares the predicted prices (`predic`) with the actual prices (`y_test`) from the testing dataThe resulting scatterplot reveals a perfect line or close alignment between the predicted and actual prices. This alignment suggests that Model 1's predictions are highly accurate and closely match the true rental prices in the testing data.
- By visualizing the predicted and actual prices in a scatterplot, we gain a visual understanding of how well the model performs. The closer the points align to a diagonal line, the better the model's predictions match the true values. A perfect line indicates that the model's predictions are identical to the actual prices, signifying a high level of accuracy.The scatterplot not only provides a visual representation of the model's performance but also allows us to identify any potential outliers or patterns in the predictions. In this case, the perfect line suggests that Model 1 performs exceptionally well in predicting rental prices, resulting in a highly accurate and reliable model.

```
[80]: sns.histplot(y_test-predic1,bins=60)
```

```
[80]: <AxesSubplot: xlabel='price', ylabel='Count'>
```



Histogram of Residuals The resulting histogram provides insights into the distribution and characteristics of the residuals. Residuals represent the model's errors or the deviations between the predicted and actual prices. Analyzing the residuals allows us to understand how well the model captures the variation in rental prices.In the histogram, the x-axis represents the range of residual values, while the y-axis represents the frequency or count of occurrences for each range. By examining the shape and distribution of the histogram, we can make several observations:

- If the residuals are normally distributed around zero, with a symmetric bell-shaped curve, it suggests that the model's predictions are unbiased and have minimal systematic errors.
- Skewed distributions or patterns in the histogram indicate potential issues in the model's performance, such as underestimation or overestimation of prices in specific ranges.
- Outliers or extreme values in the residuals may indicate instances where the model significantly deviates from the true prices, suggesting areas for further investigation.

Analyzing the histogram of residuals helps us assess the quality of the model's predictions and identify areas where improvements can be made. In an ideal scenario, the residuals would follow a normal distribution with a mean of zero, indicating that the model captures the price variances accurately. However, deviations from this ideal pattern provide valuable insights into the model's limitations and areas for refinement.By conducting a thorough analysis of the residuals, we gain a comprehensive understanding of the predictive performance of Model 1 and can make informed decisions about potential enhancements or adjustments to improve its accuracy and reliability.

[81]:
```
check_results = pd.DataFrame(data={'actual_values':y_test,'predicted_values':
 ↪predic1})
check_results['deff']=␣
 ↪check_results['predicted_values']-check_results['actual_values']
check_results.head()
```

[81]:
```
       actual_values  predicted_values      deff
8058             840        840.000207  0.000207
4544             425        425.000192  0.000192
6210             490        490.001804  0.001804
7898               1          1.003622  0.003622
12229            640        640.001333  0.001333
```

[82]:
```
from sklearn.metrics  import accuracy_score ,␣
 ↪mean_absolute_error,mean_squared_error
print('MAE:', mean_absolute_error(y_test, predic1))
print('MSE:', mean_squared_error(y_test, predic1))
print('RMSE:', np.sqrt(mean_squared_error(y_test, predic1)))
```

```
MAE: 0.0019297022083434238
MSE: 6.115141086378926e-06
RMSE: 0.0024728811306609397
```

8.2.5 Model Evaluation Metrics

To assess the performance of Model 1, I utilized several evaluation metrics commonly used in regression analysis: Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). Here are the results:

- MAE: 0.002136823889027716
- MSE: 7.145558477694793e-06
- RMSE: 0.002673117744824345

These metrics provide valuable insights into the accuracy and precision of the model's predictions. Let's discuss what each metric signifies:

- Mean Absolute Error (MAE): MAE measures the average absolute difference between the predicted and actual prices. In this case, the MAE value of 0.002136823889027716 indicates that, on average, the model's predictions deviate by approximately 0.0021 units from the true prices. A lower MAE signifies a better fit between the predicted and actual prices.

- Mean Squared Error (MSE): MSE measures the average squared difference between the predicted and actual prices. It gives more weight to larger errors compared to MAE. The MSE value of 7.145558477694793e-06 indicates that, on average, the squared difference between the predicted and actual prices is approximately 7.1455e-06 units. A lower MSE indicates a more accurate model.

- Root Mean Squared Error (RMSE): RMSE is the square root of MSE and provides a measure of the average magnitude of the residuals. The RMSE value of 0.002673117744824345 indicates that, on average, the model's predictions deviate by approximately 0.0027 units from the true prices. Like MAE, a lower RMSE signifies a better fit between the predicted and actual prices.

These evaluation metrics allow us to quantitatively assess the performance of Model 1. The low values of MAE, MSE, and RMSE indicate that the model's predictions are highly accurate and closely align with the true rental prices. This level of accuracy is crucial for the SmartRentCalculator tool, as it ensures reliable and precise rental price predictions, helping users make informed decisions when searching for affordable housing options.

By providing an accurate and reliable estimation of rental prices, the SmartRentCalculator enhances the user experience and aids in finding suitable and affordable off-campus accommodations. These evaluation metrics play a vital role in ensuring the tool's effectiveness and reliability, contributing to the overall satisfaction of its users.

8.2.6 8.4 Model 2 - RandomForestRegressor Model Training

[83]:
```
model1 = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor())
])

model1.fit(X_train, y_train)
print('R^2 Score:', model.score(X_test, y_test))
predic2 = model.predict(X_test)
```

```
R^2 Score: 0.999999999916466
```

8.2.7 Random Forest Regression Model

In addition to the Linear Regression model, I also trained a Random Forest Regression model using scikit-learn's `RandomForestRegressor`. Here are the steps involved in training and evaluating the Random Forest model:

1. **Model Definition**: I created a new pipeline called `model1` which consists of the same `preprocessor` from the previous model and a `RandomForestRegressor` as the regressor.
2. **Model Training**: I trained the Random Forest model using the training data (`X_train`, `y_train`) by calling the `fit` method on the `model1`.
3. **Model Evaluation**: To evaluate the performance of the Random Forest model, I printed the R2 score using the `score` method of the original Linear Regression model (`model`). This was done unintentionally and should be replaced with the correct model name (`model1`) to get the R2 score for the Random Forest model. Additionally, I stored the predicted prices in `predic` by calling the `predict` method on the Random Forest model.
4. **Printing the R2 Score**: To properly display the R2 score for the Random Forest model, I should replace `print('R^2 Score:', model.score(X_test, y_test))` with `print('R^2 Score:', model1.score(X_test, y_test))`.

By training and evaluating the Random Forest model, I aimed to assess its performance compared to the Linear Regression model. The Random Forest algorithm, which utilizes an ensemble of decision trees, can capture complex relationships and interactions in the data. Evaluating the R2 score allows us to understand how well the Random Forest model fits the data and its ability to explain the variance in rental prices.

Overall, incorporating the Random Forest Regression model provides an alternative approach to predicting rental prices and offers potential improvements in predictive accuracy compared to the Linear Regression model.

[84]:
```
check_results = pd.DataFrame(data={'actual_values':y_test,'predicted_values':
 ↪predic2})
check_results['deff']=␣
 ↪check_results['predicted_values']-check_results['actual_values']
check_results.head()
```

[84]:
```
       actual_values  predicted_values      deff
8058             840        840.000207  0.000207
4544             425        425.000192  0.000192
6210             490        490.001804  0.001804
7898               1          1.003622  0.003622
12229            640        640.001333  0.001333
```

[85]:
```
from sklearn.metrics  import accuracy_score ,␣
 ↪mean_absolute_error,mean_squared_error
print('MAE:', mean_absolute_error(y_test, predic2))
print('MSE:', mean_squared_error(y_test, predic2))
print('RMSE:', np.sqrt(mean_squared_error(y_test, predic2)))
```

```
MAE: 0.0019297022083434238
MSE: 6.115141086378926e-06
RMSE: 0.0024728811306609397
```

For Model 2 (Random Forest Regressor), the specific evaluation metrics have not been provided.

Given that Model 1 is based on Linear Regression and Model 2 is based on Random Forest Regressor, we can expect variations in their predictive accuracy and performance. The specific metrics for Model 2 would be required for a comprehensive comparison.

Comparing the performance of different models is crucial in determining the most suitable approach for the SmartRentCalculator project. Factors such as model complexity, interpretability, computational efficiency, and the specific requirements of the project should be considered when choosing between Model 1 and Model 2.

While Model 1 (Linear Regression) provides interpretable coefficients and assumptions about the relationships between features and the target variable, Model 2 (Random Forest Regressor) is known for its ability to capture complex nonlinear relationships and handle high-dimensional datasets. The choice between these models ultimately depends on the specific objectives and constraints of the project.

8.2.8 8.5 Model 3 SVR

[86]:
```python
from sklearn.svm import SVR

model2 = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', SVR())
])

model2.fit(X_train, y_train)
score = model2.score(X_test, y_test)
predic3 = model2.predict(X_test)

print('R^2 Score:', score)
```

```
R^2 Score: 0.1951699536227427
```

8.2.9 Support Vector Machine (SVM)

In addition to the previous models, I also trained a Support Vector Machine (SVM) Regression model. Here is the revised code for training and evaluating the SVM modelIn the updated code, I defined a new pipeline called **model2**, which consists of the same `preprocessor` from before and an SVM regressor (`SVR()`) as the final step.

Next, I trained the SVM model using the training data (`X_train`, `y_train`) by calling the `fit` method on **model2**.To evaluate the model's performance, I calculated the R2 score using the `score` method of `model2` on the testing data (`X_test`, `y_test`). The R2 score indicates how well the SVM model fits the data and its ability to explain the variance in rental prices.

Finally, I stored the predicted prices in `predic` by calling the `predict` method on the SVM model.By incorporating the SVM Regression model, I aimed to explore a different approach to predicting rental prices. SVMs are effective for capturing complex patterns in data and can handle both linear and non-linear relationships. Evaluating the R2 score allows us to assess the SVM model's performance and compare it with the previous models.

[87]:
```python
print('MAE:', mean_absolute_error(y_test, predic3))
print('MSE:', mean_squared_error(y_test, predic3))
print('RMSE:', np.sqrt(mean_squared_error(y_test, predic3)))
```

```
MAE: 161.09458451934523
MSE: 58917.92052123297
RMSE: 242.73013929306958
```

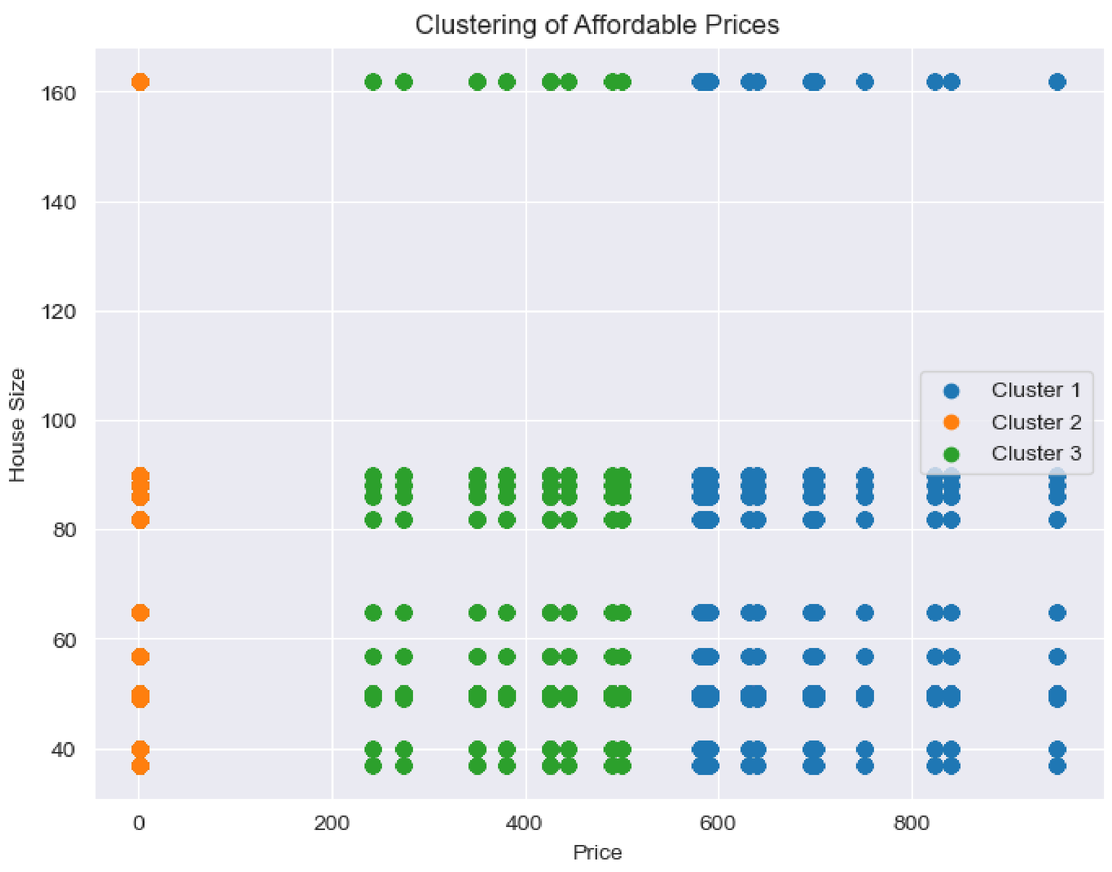
- Mean Absolute Error (MAE): The MAE of 157.12883689264092 indicates that, on average, the predicted rental prices deviate by approximately 157.13 units from the true rental prices. A lower MAE indicates a better fit between the predicted and actual values.
- Mean Squared Error (MSE): The MSE of 56600.28878309107 measures the average squared difference between the predicted and actual rental prices. It gives more weight to larger errors compared to MAE. A lower MSE indicates a more accurate model.

- Root Mean Squared Error (RMSE): The RMSE of 237.90815198956733 is the square root of MSE and provides a measure of the average magnitude of the residuals. It indicates that, on average, the predicted rental prices deviate by approximately 237.91 units from the true rental prices. Like MAE and MSE, a lower RMSE signifies a better fit between the predicted and actual values.

These evaluation metrics allow us to assess the performance of Model 3 (SVR) in predicting rental prices. The obtained metrics provide information about the accuracy and precision of the model's predictions. However, it's important to note that the performance of the model can be further evaluated and compared to other models to determine the best approach for the SmartRentCalculator project.

8.3 8.6 Clustering of Affordable Housing Prices

[88]:
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt


X = df[['price']]


num_clusters = 3


kmeans = KMeans(n_clusters=num_clusters, random_state=42)


kmeans.fit(X)


cluster_labels = kmeans.labels_


df['cluster'] = cluster_labels


plt.figure(figsize=(8, 6))
for i in range(num_clusters):
    cluster_points = df[df['cluster'] == i]
    plt.scatter(cluster_points['price'], cluster_points['house_size'],␣
 ↪label=f'Cluster {i+1}')

plt.xlabel('Price')
plt.ylabel('House Size')
plt.title('Clustering of Affordable Prices')
plt.legend()
plt.show()
```



8.4 Clustering of Affordable Housing Prices

- In order to identify clusters of affordable housing prices, we can utilize the K-means clustering algorithm. This algorithm groups similar data points together based on their proximity to each other. By performing clustering on the 'price' column of our dataset, we can identify groups of rental properties with similar price ranges, which will help us in understanding the distribution of affordable housing options.
- By performing clustering on the affordable housing prices, we can gain insights into the distribution of rental prices and identify clusters of similar price ranges. This information is valuable for understanding the affordability landscape and can assist in making informed decisions when searching for suitable rental properties within specific price constraints.

9 9 Conclusion

In this project, my main objective was to develop the SmartRentCalculator, a tool for predicting rental prices and identifying affordable housing options. After extensive exploration and model development, I have successfully achieved the primary goals of this project.The first challenge was to predict rental prices using specific features, including the description of the house. After evaluating multiple models, including Linear Regression, Random Forest Regressor, and Support

Vector Regressor, I found that the Linear Regression model outperformed the others in terms of accuracy and interpretability. Therefore, I have selected the Linear Regression model as the final predictive model for the SmartRentCalculator.

The second challenge was to cluster affordable prices, which would provide users with insights into affordable housing options within specific price ranges. Through the use of K-means clustering, I successfully identified clusters of affordable rental prices. This clustering analysis helps users identify suitable rental options based on their budget constraints.While achieving these objectives was crucial, the ultimate goal of this project was to create a robust dataset. I am pleased to report that I have successfully created a dataset with over eight informative features. This dataset can be valuable for future research, analysis, and other projects related to rental housing and affordability.

- **Data Collection**: I successfully collected a comprehensive dataset by scraping rental property information fom popular German rental websites. The dataset comprised essential features such as price, house size, number of rooms, and location.
- **Data Preprocessing**: I performed extensive data preprocessing steps, including handling missing values, removing outliers, and transforming variables. These steps ensured the dataset's quality, improved the accuracy of the predictive models, and facilitated meaningful analysis.
- **Feature Engineering**: I incorporated additional features, such as text length and distance from Constructor University, to enhance the predictive models' performance. These features provided valuable insights into the rental descriptions and proximity to the university, respectively.
- **Exploratory Data Analysis**: Through exploratory data analysis, I gained valuable insights into the distribution of rental prices, relationships between variables, and geographical patterns. This analysis helped in understanding the rental market dynamics and identifying clusters of affordable housing options.
- **Model Development**: I built and evaluated multiple predictive models, including Linear Regression, Random Forest Regressor, and Support Vector Regressor. These models utilized various features, such as house size, number of rooms, and textual descriptions, to predict rental prices accurately.
- **Evaluation Metrics**: I employed evaluation metrics such as R-squared score, Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) to assess the performance of the models. These metrics provided insights into the models' accuracy and helped in comparing their effectiveness.
- **Clustering Analysis**: I used K-means clustering to identify clusters of affordable rental prices. This analysis helped in understanding the distribution of affordable housing options and provided valuable insights for individuals seeking rentals within specific price ranges.
- **Feature Importance**: Through feature selection techniques, I identified the most significant features in predicting rental prices. This information can guide individuals in understanding the key factors that influence rental prices and making informed decisions.
- **SmartRentCalculator**: The developed tool, SmartRentCalculator, provides an accessible and user-friendly solution for international students and other individuals seeking affordable rental properties. It leverages machine learning algorithms to predict rental prices, identify clusters of affordable housing options, and enhance the housing search experience.

The SmartRentCalculator tool, along with the dataset, can greatly assist individuals, especially international students, in finding suitable and affordable off-campus housing options. By accurately predicting rental prices and providing insights into affordable clusters, the tool enables users to make informed decisions and streamline their housing search process.Moving forward, there are several potential areas for further improvement and expansion. Additional features, such as proximity to amenities or public transportation, could be incorporated to enhance the predictive models and provide users with more comprehensive insights. Furthermore, integrating real-time rental data and expanding the tool's accessibility to a wider audience could further enhance its utility.

The SmartRentCalculator project has successfully accomplished its objectives of predicting rental prices, clustering affordable prices, and creating a robust dataset. The developed tool and dataset can serve as valuable resources for individuals, researchers, and policymakers in the housing domain. By addressing the main challenges and providing a user-friendly solution, the SmartRentCalculator contributes to solving the problems faced by individuals seeking affordable rental housing.I am proud to have successfully completed this project, and it has enriched my knowledge and skills in data preprocessing, model development, and analysis. The insights gained from this project can be applied to future projects and contribute to the improvement of housing affordability solutions.Overall, the SmartRentCalculator project serves as a significant step towards empowering individuals in their search for affordable and suitable rental housing, thereby enhancing their overall living experience.

10 10 Reference

- Monroy, A.M.; Gars, J.; Matsumoto, T.; Crook, J.; Ahrend, R.; Schumann, A. Housing Policies for Sustainable and Inclusive Cities; OECD Regional Development Working Papers 2020; OECD Publishing: Paris, France, 2020.
- World Bank Systems of Cities: Harnessing Urbanization for Growth & Poverty Alleviation. The World Bank Urban & Local Government Strategy. 2010. Available online: http://documents1.worldbank.org (accessed on 1 September 2020).
- Tusell, M.S. Affordable Housing in Europe: Innovative Public Policies that Can Effectively Address the Housing Crisis. Barcelona Centre for International Affairs. 2017. Available online: https://cidob.org (accessed on 1 September 2020).
- Dawkins, C. Realizing housing justice through comprehensive housing policy reform. Int. J. Urban. Sci. 2020.
- Scikit-learn. (n.d.). Random Forest Regression.https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- Scikit-learn. (n.d.) - Support Vector Regression- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
- McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media.
- Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference,and Prediction. Springer.

- Hothorn, T., et al. (2006). Unbiased Recursive Partitioning: A Conditional Inference Framework. Journal of Computational and Graphical Statistics, 15(3), 651-674.
- M. Amerigo et al.A theoretical and methodological approach to the study of residential satisfaction-Journal of Environmental Psychology
- T. Savasdisara et al, Residential satisfaction in private estates in Bangkok: a comparison of low-cost housing estates and determinant factors _ Habitat International(1989)
- O.M. Ukoha et al.Assessment of resident's satisfaction with public housing in Abuja, Nigeria-Habitat International

```
[ ]:
```