DAT565/DIT407 Assignment 2

Saif Sayed
gussayedfa@student.gu.se

Gona Ibrahim Abdulrahman
gusibrgo@student.gu.se

2024-04-10

This is a report for assignment 2 for the course *Introduction to Data Science & AI* from Chalmers and Gothenburg University.

Problem 1: Scraping house prices

In this task, the goal was to scrape house prices from a tarball file containing HTML files and extract relevant information from each advertisement of a closing price. The data was scraped from Hemnet in October 2023. The extracted information included the date of sale, address, location, bo-area (habitable area), bi-area (non-habitable area), total-area (combined area), rooms, area of the plot, and the closing price.

To solve the problem, the following steps were followed:

Importing the necessary libraries: The code began by importing the required libraries, including os, tarfile, pandas, datetime, re, and BeautifulSoup.

Extracting the HTML files: The tarball file was opened, and the HTML files were extracted to a designated directory. The list of HTML files was obtained using the os module.

Parsing HTML files: The code then looped through each HTML file, opened it, and read its contents. Beautiful Soup was used to parse the HTML content.

Extracting information from advertisements: Within each HTML file, the code used Beautiful Soup to find all advertisements. For each advertisement, the relevant information was extracted, including the date of sale, address, location, bo-area, bi-area, total-area, rooms, area of the plot, and the closing price. Discrepancies in how the information was presented were taken into account.

Storing the extracted data: The extracted information was stored in a list as individual entries. The information for each advertisement was appended to the list.

Converting data to a DataFrame and saving as CSV: The data list was converted to a pandas DataFrame. Finally, the DataFrame was saved as a CSV file named "*house_prices.csv*" with the specified columns.

The resulting CSV file contains the extracted information for each entry in the dataset, including the address, location, bo-area, bi-area, total-area, rooms, area of the plot, date of sale, and the sale price.

By following these steps, the problem of scraping house prices and extracting relevant information from the HTML files was successfully solved. The resulting CSV file can be further analyzed and used for various purposes like data exploration, visualization, or modeling.

Our source code can be found in Appendix A of this document.

Problem 2: Analyzing 2022 house sales

In this task, we analyzed the data of houses sold in 2022 and produced several plots to gain insights into the closing prices and their relationship with other variables. The steps taken to analyze the data and create the plots are described in the following sub-sections.

1 Five Number Summary

Filtering the data: We created a new DataFrame called *houses_sold_*2022 by selecting houses that were sold in 2022 from the original dataset.

Computing the five-number summary: We computed the five-number summary (minimum, maximum, median, first quartile, and third quartile) of the closing prices for houses sold in 2022 using the describe() function on the 'Sale Price (kr)' column (see Figure 1).

```
Minimum:  1650000.0  kr
Maximum:  10500000.0  kr
Median:  5000000.0  kr
First Quartile:  4012500.0  kr
Third Quartile:  5795000.0  kr
```

Figure 1: Five number summary for houses sold in 2022

2 Histogram

Histogram of closing prices: We constructed a histogram to visualize the distribution of closing prices. The number of bins was determined using the square root choice method, where the number of bins is the square root of the total number of observations (see Figure 2).

3 Scatter Plot

Scatter plot of closing price vs. bo-area: We created a scatter plot to examine the relationship between the closing price and the boarea (habitable area) of the houses sold in 2022 (see Figure 3).

4 Scatter Plot (grouped by Number of Rooms)

Scatter plot with color-coded rooms: We repeated the scatter plot but colorized the observations based on the number of rooms in each house. Each point in the scatter plot was assigned a color based on the number of rooms, using a colormap (see Figure 4).

5 Discussion

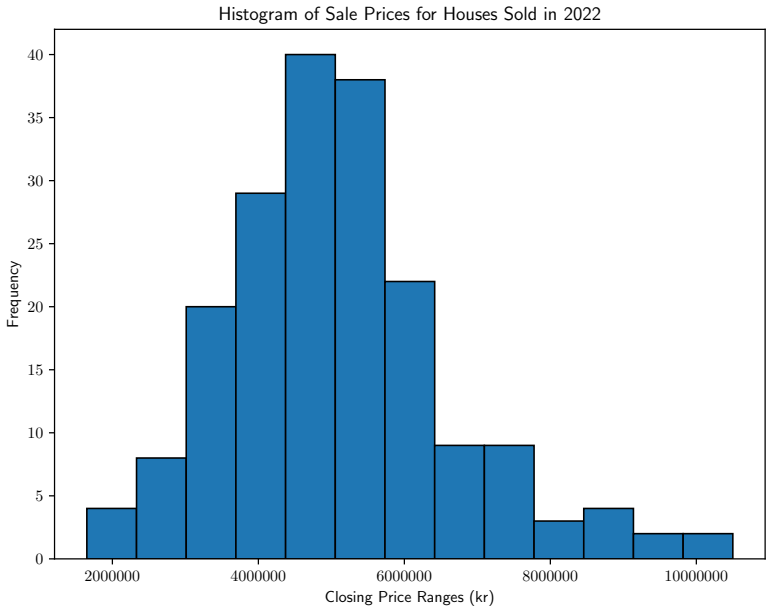

Figure 2: Frequency of Sale Prices for Houses Sold in 2022

Figure 2: The histogram in the image represents the distribution of sale prices for houses sold in 2022.

The histogram shows the distribution of house sale prices in 2022. Most houses were sold in the price range of 400 000 kr to 600 000 kr, with a peak frequency around 35. As prices increased beyond 600 000 kr, the frequency of sales declined. Below 400 000 kr, there was a sharp drop in frequency. This suggests a price sensitivity among buyers. However, further analysis would be needed to understand other factors influencing house prices.

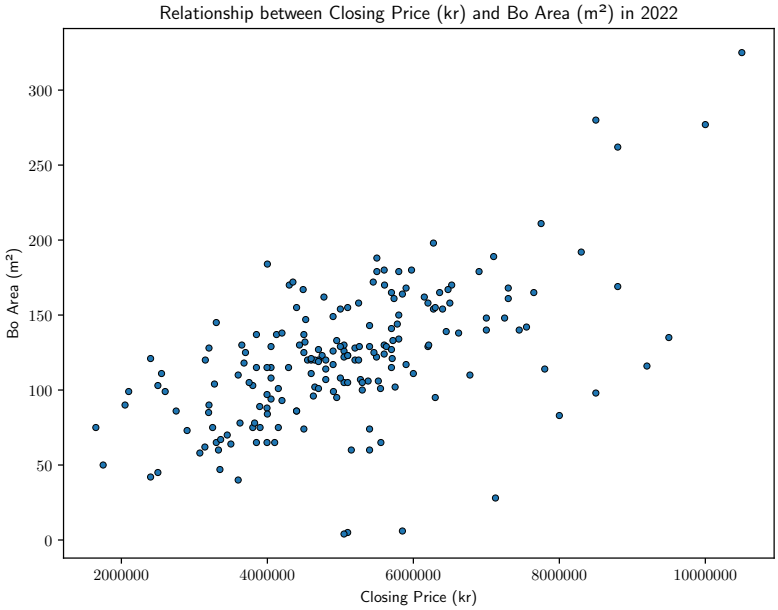

Figure 3: Relationship between Closing Price (kr) and Bo Area (m^2) in 2022

Figure 3 and Figure 4: The scatter plots illustrate the relationship between Closing Price (kr) and Bo Area (m^2) for houses sold in 2022:

The scatter plot shows that larger Bo Area (m^2) tends to be associated with higher Closing Prices (kr) for houses sold in 2022. Buyers are willing to pay more for spacious properties. However, there is price variability even for similar area sizes. Other factors like location and amenities also play a role. Overall, the positive correlation emphasizes that size matters in house prices.

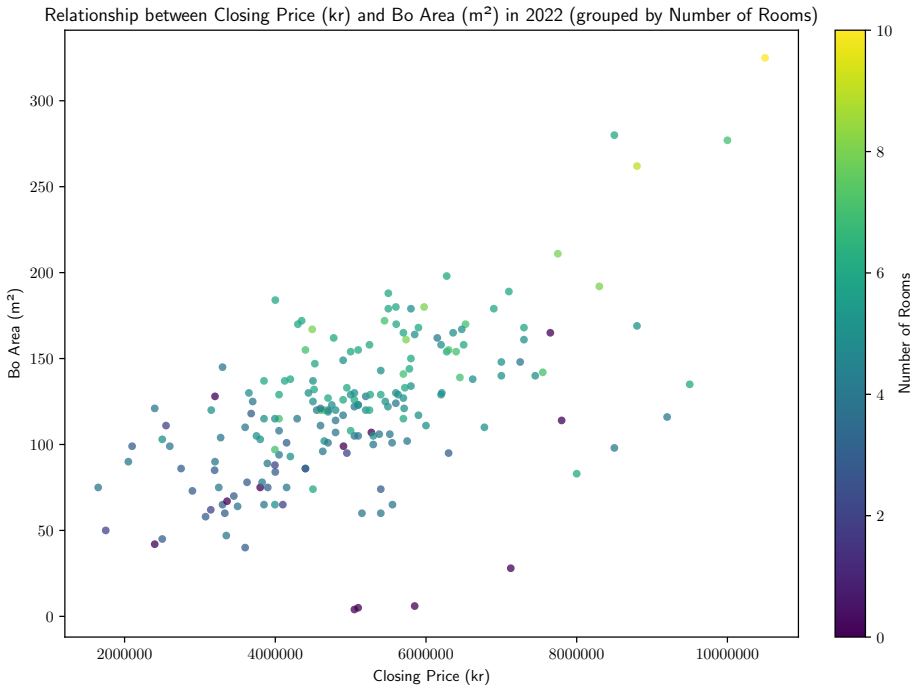

Figure 4: Relationship between Closing Price (kr) and Bo Area (m^2) in 2022 (grouped by Number of Rooms)

A Python code

This is the code we used to extract relevant information, perform data cleaning and plot our figures.

```
#import modules
import re
import csv
from bs4 import BeautifulSoup
import os
import tarfile
import pandas as pd
import locale
from datetime import datetime
import numpy as np
import matplotlib.pyplot as plt

#set local datetime to swedish to interpret swedish
    months
locale.setlocale(locale.LC_TIME, "sv_SE")
'''
Problem 1
'''
```

```
# Get current working directory of source file
root_dir = os.getcwd()

# Path to the input tar.gz file
tar_file_path = os.path.join(root_dir, '
    kungalv_slutpriser.tar.gz')
# Path where the tarbile file is extracted to
extract_dir = root_dir
# Path to all the html files
html_dir = os.path.join(root_dir, 'kungalv_slutpriser'
    )

# open tarbile for reading
with tarfile.open(tar_file_path, 'r', encoding='utf-8'
    ) as tar:
    tar.extractall(extract_dir) # extract all files
    file_list = os.listdir(html_dir) # list all html
        files in directory
    data = [] # create placeholder to store data
    # loop through all the html files
    for filename in file_list :
        # Check if the extracted file is an HTML file
        if filename.endswith('.html'):
            file_path = os.path.join(html_dir,
                filename)
            #open a specific html file in the above
                file_path
            with open(file_path, 'r', encoding='utf-8'
                ) as file:
                # Read the contents of the HTML file
                html_content = file.read()
            # Parse html file using beautifulsoup
            soup = BeautifulSoup(html_content, 'html.
                parser')
            # find all ads by html id/class
            ads = soup.find_all('li', class_='sold-
                results__normal-hit')
            # loop through all the ads
            for ad in ads:
            # Initialize variables for each piece of
                information
                date_of_sale = ''
                address = ''
                location = ''
                bo_area = ''
                bi_area = ''
                total_area = ''
                area_plot = ''
                sale_price = ''
                rooms = ''
```

```
58
59                  # Extract the date of sale, find
                        element by id/class
60                  date_elem = ad.find('span', class_='
                        hcl-label hcl-label--state hcl-
                        label--sold-at')
61                  if date_elem:
62                      date_of_sale = date_elem.text.
                            strip().replace("S ld ", "")
63                      date_of_sale = datetime.strptime(
                            date_of_sale, '%d %B %Y')
64                      date_of_sale = pd.to_datetime(
                            date_of_sale, format = '%Y-%m-%
                            d') # converts to datetime
                            format
65
66                  # Extract the address and location,
                        find element by id/class
67                  address_elem = ad.find('h2', class_='
                        sold-property-listing__heading qa-
                        selling-price-title hcl-card__title
                        ')
68                  if address_elem:
69                      address = address_elem.text.strip
                            ()
70
71                  parent_location = ad.find('span',
                        class_='property-icon property-icon
                        --result').parent
72                  if parent_location:
73                      parent_location.span.decompose()
74                      location_strip = parent_location.
                            text.strip().replace("\n", "")
                            # remove unnecessary string
                            literals
75                      location_split = location_strip.
                            split(',')
76                      locations = list(map(lambda x: x.
                            strip(), location_split)) #
                            remove leading white spaces
77                      location =  ", ".join(locations)
78
79                  # Extract the bi area, find element by
                        id/class
80                  bi_elem = ad.find('span', class_='
                        listing-card__attribute--normal-
                        weight')
81                  if bi_elem:
82                      bi_text = bi_elem.text.strip()
```

```
83                      match = re.search(r'(\d+)',
                            bi_text) # collect only the
                            digits using regex in the
                            extracted data
84                      if match:
85                          bi_area = match.group(1)
86                      else:
87                          bi_area= ""
88
89                  # Extract the bo area, bi area, rooms
                        and total area; find element by
                        id/class
90                  area_elem = ad.find('div', class_='
                        sold-property-listing__subheading␣
                        sold-property-listing__area')
91                  if area_elem:
92                      area_elem.span.decompose if
                            area_elem.span else area_elem
93                      area_text = area_elem.text.strip()
94                      match = re.findall(r'\d+',
                            area_text) # collect only the
                            digits using regex in the
                            extracted data
95                      if len(match) > 2:
96                          # if 3 pieces of info.
                                available for area and room
                                , then store all three data
                                 and compute total area.
97                          bo_area = int(match[0])
98                          bi_area = int(match[1])
99                          rooms = int(match[2])
100                         total_area = int(match[0]) +
                                int(match[1])
101                     elif len(match) == 2 and bi_area
                            != "":
102                         # if 2 pieces of info.
                                available which includes bi
                                 area, then set room to 0,
                                calculate total area
103                         bo_area = int(match[0])
104                         bi_area = int(match[1])
105                         rooms = 0
106                         total_area =  int(match[0]) +
                                int(match[1])
107                     elif len(match) == 1:
108                         # if only 1 piece of info.
                                available, then set room
                                and bi area to 0, calculate
                                 total area
109                         bo_area = int(match[0])
```

```
110                         bi_area = 0
111                         rooms = 0
112                         total_area =  int(match[0])
113                     elif len(match) == 2 and bi_area
                            == "":
114                         # if 2 pieces of info.
                                available which doesn't
                                include bi area, then set
                                bi area to 0
115                         bo_area = int(match[0])
116                         bi_area = 0
117                         rooms = int(match[1])
118                         total_area =  int(match[0])
119
120                 # Extract the area plot, find element
                        by id/class
121                 plot_elem = ad.find('div', class_='
                        sold-property-listing__land-area')
122                 if plot_elem:
123                     plot_text = plot_elem.text.strip()
124                     match = re.findall(r'\d+',
                            plot_text) # collect only the
                            digits using regex in the
                            extracted data
125                     if match:
126                         area_plot = str(match[0]) + "␣
                                m  " + "␣tomt"
127
128                 # Extract the sale price, find element
                        by id/class
129                 price_elem = ad.find('span', class_='
                        hcl-text␣hcl-text--medium')
130                 if price_elem:
131                     price = price_elem.text.strip()
132                     match = re.findall(r'\d+', price)
                            # collect only the digits using
                             regex in the extracted data
133                     if match:
134                         sale_price = ""
135                         for value in match:
136                             sale_price += str(value)
137                         sale_price = int(sale_price)
138                 # Append the extracted data to the
                        list
139                 data.append([address, location,
                        bo_area, bi_area, total_area, rooms
                        , area_plot, date_of_sale,
                        sale_price])
140 #store the data in a panda dataframe
```

```
df = pd.DataFrame(data, columns=['Address', 'Location', 'Bo Area ( m  )', 'Bi Area ( m  )', 'Total Area ( m  )', 'Rooms', 'Area Plot', 'Date of Sale', 'Sale Price (kr)'])
df.to_csv('house_prices.csv', index=False, encoding='utf-8') # convert to csv

'''
Problem 2
'''
houses_sold_2022 = df[df['Date of Sale'].dt.year == 2022] #create and store dataframe only for the year 2022
sale_price_stats = houses_sold_2022['Sale Price (kr)'].describe()
#collect statistical data about sales price on the dataframe
minimum = sale_price_stats['min']
maximum = sale_price_stats['max']
median = sale_price_stats['50%']
first_quartile = sale_price_stats['25%']
third_quartile = sale_price_stats['75%']

print("Minimum: ", minimum, " kr")
print("Maximum: ", maximum, " kr")
print("Median: ", median, " kr")
print("First Quartile: ", first_quartile, " kr")
print("Third Quartile: ", third_quartile, " kr")

'''Histogram of Sale Prices for Houses Sold in 2022'''

sale_prices = np.asarray(houses_sold_2022['Sale Price (kr)'].dropna(), float)
fig1, ax1 = plt.subplots(figsize=(8, 6))
# the number of bins was determined using the square root choice method
num_bins = int(len(houses_sold_2022['Sale Price (kr)']) ** 0.5)
ax1.hist(sale_prices, bins=num_bins, edgecolor='black', linewidth=1, label='Closing prices')
ax1.set_xlabel('Closing Price Ranges (kr)')
ax1.set_ylabel('Frequency')
ax1.set_title('Histogram of Sale Prices for Houses Sold in 2022')
ax1.ticklabel_format(useOffset=1, style='plain', axis='x') # change the tick format on the x-axis to be more comprehensive
fig1.savefig('histogram.pdf')

```

```
"""Relationship between Closing Price (kr) and Bo Area
    (m  ) in 2022"""

fig2, ax2 = plt.subplots(figsize=(8, 6))
bo_area = pd.to_numeric(houses_sold_2022['Bo Area (m
    )'].dropna(), errors ='coerce').values
ax2.scatter(sale_prices, bo_area, s= 15, edgecolor='
    black', linewidth=0.5)
ax2.set_title("Relationship between Closing Price (kr)
    and Bo Area (m  ) in 2022")
ax2.set_xlabel('Closing Price (kr)')
ax2.set_ylabel('Bo Area (m  )')
ax2.ticklabel_format(useOffset=1, style='plain', axis=
    'x') # change the tick format on the x-axis to be
    more comprehensive
fig2.savefig('scatter_plot.pdf')

'''Relationship between Closing Price (kr) and Bo Area
    (m  ) in 2022 (grouped by Number of Rooms)'''

fig3, ax3 = plt.subplots(figsize=(8, 6),
    constrained_layout=True)
rooms = pd.to_numeric(houses_sold_2022['Rooms'].dropna
    (), errors ='coerce').values
ax3.scatter(sale_prices, bo_area , c= rooms, cmap='
    viridis', s=15, alpha= 0.75)
ax3.set_title("Relationship between Closing Price (kr)
    and Bo Area (m  ) in 2022 (grouped by Number of
    Rooms)")
ax3.set_xlabel('Closing Price (kr)')
ax3.set_ylabel('Bo Area (m  )')
sm = plt.cm.ScalarMappable(cmap='viridis') # mapping
    scalar values to the colormap based on the values
    stored in 'rooms'
sm.set_array(rooms)
fig3.colorbar(sm, label='Number of Rooms', ax=ax3) #
    create colorbar
ax3.ticklabel_format(useOffset=1, style='plain', axis=
    'x') # change the tick format on the x-axis to be
    more comprehensive
fig3.savefig('scatter_plot_room.pdf')
```