DIT407 Introduction to data science and AI - Assignment 2

Reynir Siik
reynir.siik@gmail.com

Franco Zambon
guszamfr@student.gu.se

2024-01-31

Abstract

In this assignment, we practice data cleaning by extracting the interesting data from a source that contains the data, not quite in the format we want. In the second part, we then produce some plots about the data.

1 Problem 1: Scraping house prices

1.1 Introduction

Abbreviated citation from assignment:

The file `kungalv slutpriser.tar.gz` contains the closing prices of houses, sold in Kungälv municipality between 2013 and 2023. The data has been scraped from [1] in October 2023. Your task will be to convert the data into a form that can be analyzed. The tarball contains a number of HTML files. Use Beautiful Soup2 to parse the HTML and extract the relevant pieces of information for each advertisement of a closing price.

The data of interest is: date of sale, address , location, building living area, building utility area, number of rooms, plot area and closing price.

The data consists of closed listings that is generated as the web page is loaded into the browser. There is no guarantee that every value is present, and some values can have a decimal value.

The `html` data from every file is read into an object `soup` by the module `BeautifulSoup`.

```
while i<40:
    i += 1
    print("File␣", i)

    fileSequence = str(i).rjust(2, '0')
    with open(
        "Uppgift02\\code\\kungalv_slutpriser\\kungalv_slutpris_page_"
        + fileSequence
        + ".html",
        encoding="utf8") as fp:
        soup = BeautifulSoup(fp, 'html.parser')
```

Listing 1: Reading the files

1.2 Overview of dataset

Every listing is contained in a separate list item which makes it easy to filter them out into separate items. With `soup.findall` the listings are filtered out into a new object `p` as separate items.

```
## Divide into sold listings ############################################
    p = soup.find_all('li', class_='sold-results__normal-hit')
```

Listing 2: Filtering out the listings

Within every item most data is contained under specific ”labels”. The most challenging values to retrieve is building living area, building utility area and number of rooms, since they are not separated by any labels. The values may, or may not be present, and they can also consist of decimal values.

```
        tag = t.find(
            'div',
            class_="sold-property-listing__subheading␣"
            "sold-property-listing__area")
```

Listing 3: Getting the area section

1.3 Retrieving building living area, building utility area and number of rooms

With regex all digit values are retrieved into a list variable.

```
        pArea = re.findall(r"\d+[,]?\d?",  tag.get_text(strip=True))
```

Listing 4: Getting the area values into a list

By looking at ”telltales” `[+]` and `[rum]` the data in the list is identified. Data is assigned to the relevant variable and decimal comma is changed to decimal point before writing it to csv-file.

```
        bia = re.search(r"[+]", tag.get_text(strip=True))
        rum = re.search(r"rum", tag.get_text(strip=True))
```

Listing 5: Analyzing presence of data

1.4 Transforming the date field

The closing date is written in the format; `Såld 21 juni 2023`. The month is in Swedish and does not work well with `datetime`. By changing month to English and three letter abbreviation the problem is solved.

```
        ## Transform the date
        ## This could be a function... I know...
        monthdict =     {
            "januari": "Jan",
            "februari": "Feb",
            "mars": "Mar",

            "december": "Dec",
        }

        monthlist = [
            "januari",
            "februari",
            "oktober",
            "november",
            "december"
        ]
        for month in monthlist:
            if date_of_sale.find(month)>0:
                date_of_sale = date_of_sale.replace(
                    month, monthdict[month])
        #####################
```

Listing 6: Transforming date

1.5 Writing the cvs file

By opening a file for writing and make use of the `cvs` module, writing a cvs file is fairly trivial. The file is opened in the beginning of the code, and a `cvs.writer` object with the file descriptor is defined. The first row with the headings is written to the file.

```
csvfile = open(csv_file_path, 'w', newline='')
writer = csv.writer(csvfile)
```

Listing 7: Opening CVS file

When all values are retrieved for an individual listing the data is written to the file by composing a list with the values, and writing it to the file with the `writer` object. Finally the file is closed.

```
        ## Put the data in the csv file
        myList = (
            datetime.datetime.strptime(date_of_sale, r"%d␣%b␣%Y").date(),
            PropAddress,
            Location_of_the_estate,
            Area_boarea,
            Area_biarea,
            The_number_of_rooms,
            Area_of_the_plot,
            Closing_price)
        writer.writerow(myList)
```

Listing 8: Writing the CVS file

2 Problem 2: Analyzing 2022 house sales

2.1 Introduction

Abbreviated citation from assignment:

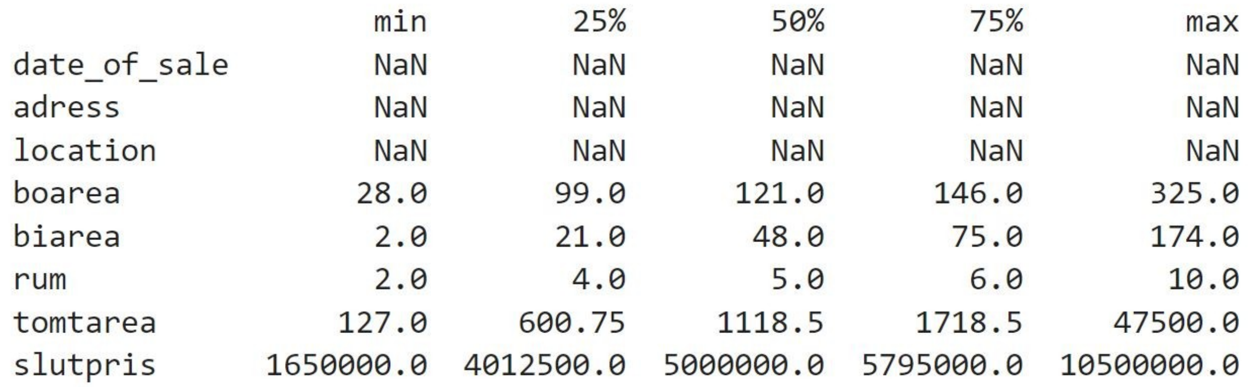
We will now analyze the data, and produce some plots. Select all houses that were sold in 2022, and answer all points below. Include the plots you create in your report and the answers to any other questions.

2.2 The five-numbers summary

The five-number summary is a set of descriptive statistics that provides information about a dataset. It consists of the five most important sample percentiles:

- the sample minimum (smallest observation)
- the lower quartile or first quartile
- the median (the middle value)
- the upper quartile or third quartile
- the sample maximum (largest observation)

In order for these statistics to exist, the observations must be from a univariate variable that can be measured on an ordinal, interval or ratio scale.[2]
You can see the five-numbers summary that was generated selecting all the houses that got sold in 2022 in figure 1.



```
                    min        25%        50%        75%         max
date_of_sale        NaN        NaN        NaN        NaN         NaN
adress              NaN        NaN        NaN        NaN         NaN
location            NaN        NaN        NaN        NaN         NaN
boarea             28.0       99.0      121.0      146.0       325.0
biarea              2.0       21.0       48.0       75.0       174.0
rum                 2.0        4.0        5.0        6.0        10.0
tomtarea          127.0     600.75     1118.5     1718.5     47500.0
slutpris      1650000.0  4012500.0  5000000.0  5795000.0  10500000.0
```

Figure 1: Five-numbers summary

From this summary we can see how 50% of the houses has between 4 and 6 rooms, a price that ranges between 4 and 5.8 millions SEK and an inhabitable area between 99 and 146 sqm, a non-inhabitable area between 21 and 75 sqm and a plot area between 600 and 1700 sqm. Besides that there could be some house with extreme values, such as one with a plot area of 47500 sqm.

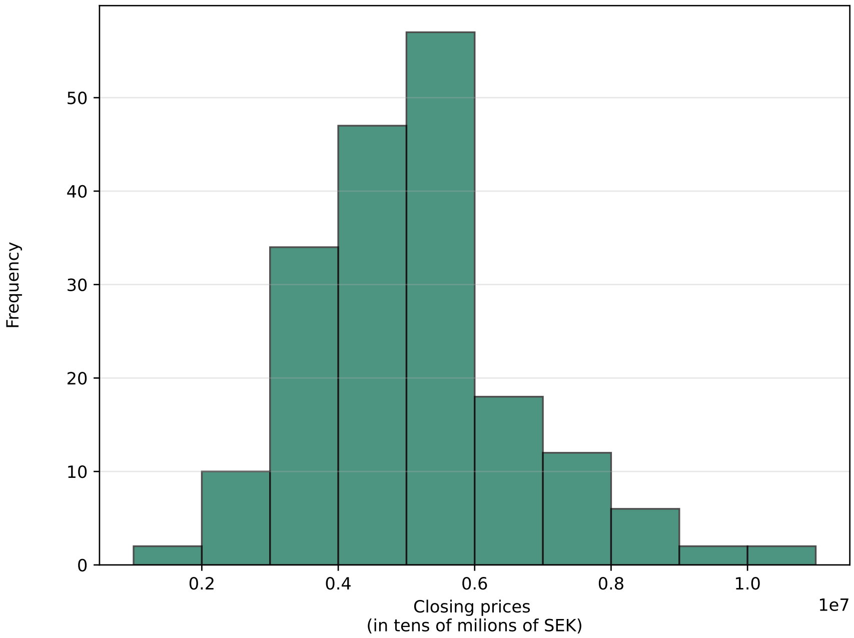


Figure 2: Distribution of closing prices in 2022

2.3 The histogram

Plotting the closing prices of all the houses sold in 2022 we obtained the histogram of figure 2.

The number of bins has been chosen to avoid over- or under-fitting the data and each bin was chosen to start and to end with a round number to increase readability. The histogram suggests normally-distributed data with a heavier right tail.

2.4 The prices vs area scatter plot

In figure 3 we plotted the closing prices and the boarea (the total inhabitable area) of the houses sold in 2022.

In this graph we could see how, as expected, the boarea is linearly correlated to the closing price. However there's some house which seems to have a price very high compared to its boarea (for example the house that has an area lower than 50 sqm but a price higher than 7 millions SEK) and this is probably caused by other factors such as the position of the house.

2.5 The prices vs area scatter plot by number of rooms

Finally in figure 4 we plotted the same graph of above but this time we distinguished the houses by the number of rooms. The color of the dots helps doing that.

As expected we can see in this graph how the houses with higher boareas tend to have more rooms.

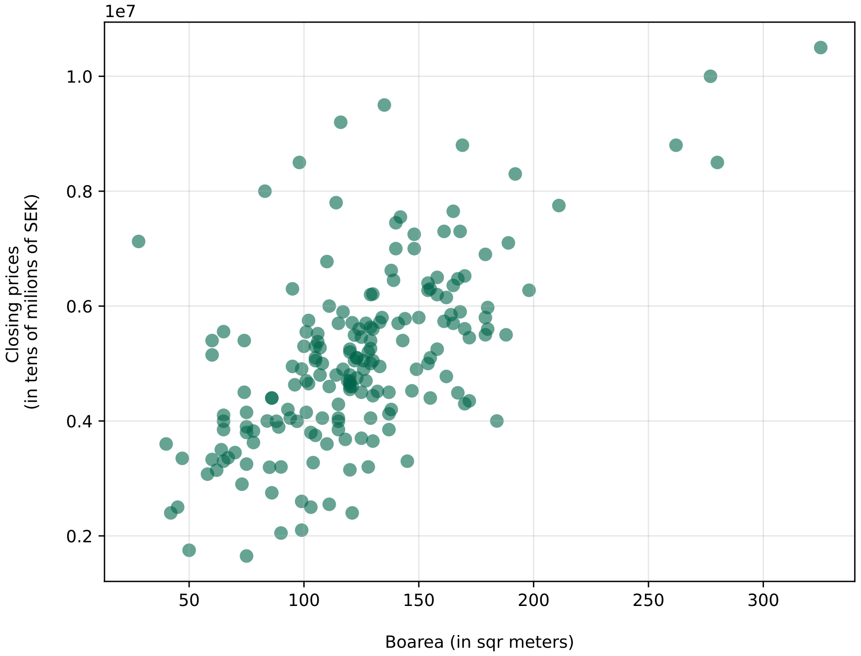


Figure 3: Scatter plot of the boarea and the closing prices

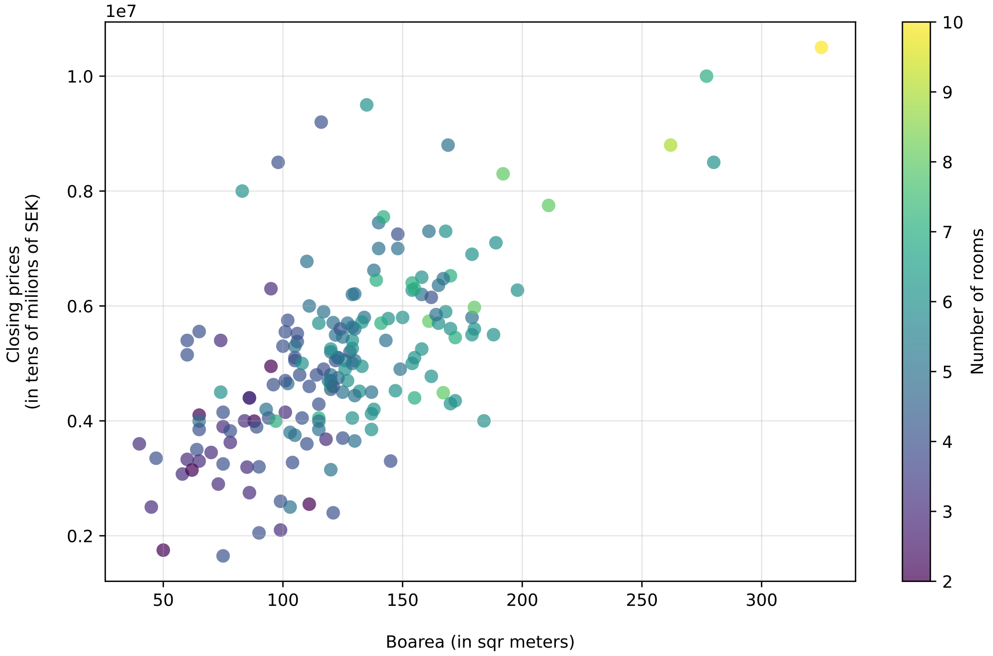


Figure 4: Scatter plot of the boarea and the closing prices differentiated by number of rooms

References

[1] Hemnet, *Slutpriser*, Made available on Canvas, From assignment handout on Canvas, Oct. 2023. [Online]. Available: `https://www.hemnet.se/salda/bostader?location_ids%5B%5D=17973&item_types%5B%5D=villa`.

[2] *Five-number summary*, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Five-number_summary.

A Complete source code

This is the complete listing of the source code.

```
# Reynir Siik, Franco Zambon
# DIT407 lp3 2024-01-28
# Assignment 2
# Problem 1 and 2
# Scraping Hemnet
#
## #####################################################################
import datetime
from math import nan
from os import replace
from bs4 import BeautifulSoup
import re
from matplotlib.dates import date2num
import matplotlib.pyplot as plt
import numpy  as np
import pandas as pd
import csv
## #####################################################################

csv_file_path = "Uppgift02\\code\\huspriser.csv"

csvfile = open(csv_file_path, ’w’, newline=’’)
writer = csv.writer(csvfile)
writer.writerow([
    ’date_of_sale’,
    ’adress’,
    ’location’,
    ’boarea’,
    ’biarea’,
    ’rum’,
    ’tomtarea’,
    ’slutpris’])

i = 0

while i<40:
    i += 1
    print("File␣", i)

    fileSequence = str(i).rjust(2, ’0’)
    with open(
        "Uppgift02\\code\\kungalv_slutpriser\\kungalv_slutpris_page_"
        + fileSequence
        + ".html",
        encoding="utf8") as fp:
        soup = BeautifulSoup(fp, ’html.parser’)
    fp.close()

## Divide into sold listings ###########################################
    p = soup.find_all(’li’, class_=’sold-results__normal-hit’)

## Extract #############################################################
    for t in p:
        ## Date #############
        tag = t.find(
            ’span’,
            class_="hcl-label␣hcl-label--state␣hcl-label--sold-at")
        lbl = re.findall(
            r"\d?\d{1}\s\w+\s\d{4}",
            tag.get_text(strip=True))

        date_of_sale = str(lbl).strip("’[]")
        tag.clear()

        ## Address ##########
        tag = t.find(
            ’h2’,
            class_="sold-property-listing__heading␣"
            "qa-selling-price-title␣hcl-card__title")
        PropAddress = str(
            re.findall(r".+", tag.get_text(strip=True))).strip("’[]")
        tag.clear()

        ## Remove tag that will clutter the data
        tag = t.find(’span’,
                     class_="property-icon␣property-icon--result")
        tag.clear()

        ## Location #########
        tag = t.find(’div’, class_="sold-property-listing__location")
        Location_of_the_estate = str(
            re.sub(r"\s+", "_",
                   str(re.findall(r".+", tag.get_text(strip=True)))))
        Location_of_the_estate = str(
            re.sub("_", "␣",
                   re.sub("__",",␣",
                          re.sub(
                              r"[,][’][,]|\W",
                              "",
                              Location_of_the_estate))))
        tag.clear()

        ## Number of rooms, boarea, biarea ##
        tag = t.find(
            ’div’,
            class_="sold-property-listing__subheading␣"
            "sold-property-listing__area")

        pArea = re.findall(r"\d+[,]?\d?",  tag.get_text(strip=True))
        lenP = len(pArea)

        bia = re.search(r"[+]", tag.get_text(strip=True))
        rum = re.search(r"rum", tag.get_text(strip=True))

        bia_t = not(bia == None)
        rum_t = not(rum == None)
        x = (lenP, bia_t, rum_t)
        match x:
            case (3, True, True):
                Area_boarea = pArea[0]
                Area_biarea = pArea[1]
                The_number_of_rooms = pArea[2]
            case (2, False, True):
                Area_boarea = pArea[0]
                Area_biarea = np.NaN
                The_number_of_rooms = pArea[1]
            case (2, True, True):
                Area_boarea = np.NaN
                Area_biarea = pArea[0]
                The_number_of_rooms = pArea[1]
            case (2, True, False):
                Area_boarea = pArea[0]
                Area_biarea = pArea[1]
                The_number_of_rooms = np.NaN
            case (1, False, False):
                Area_boarea = pArea[0]
                Area_biarea = np.NaN
                The_number_of_rooms = np.NaN
            case (1, False, True):
                Area_boarea = np.NaN
                Area_biarea = np.NaN
                The_number_of_rooms = pArea[0]
            case (1, True, False):
                Area_boarea = np.NaN
                Area_biarea = pArea[0]
                The_number_of_rooms = np.NaN
            case (0, bia_t, rum_t):
                Area_boarea = np.NaN
                Area_biarea = np.NaN
                The_number_of_rooms = np.NaN
            case _:
                Area_boarea = np.NaN
                Area_biarea = np.NaN
                The_number_of_rooms = np.NaN
                print('ERROR␣ERROR␣ERROR␣ERROR␣ERROR␣ERROR␣ERROR␣')

        ## Decimal comma to decimal point
        Area_boarea = re.sub("[,]", "." ,str(Area_boarea))
        Area_biarea = re.sub("[,]", "." ,str(Area_biarea))
        The_number_of_rooms = re.sub("[,]", "." ,str(The_number_of_rooms))

        ## Area of the plot #
        tag = t.find(
            'div',
            class_="sold-property-listing__land-area")
        if len(str(tag))>4:
            lbl = str(
                re.findall(r".+",
                           tag.get_text(strip=True).strip(r"'[]")))
            Area_of_the_plot = str(
                re.findall(r"\d+",
                           re.sub(r"\W*|[x][a][0]|[t]{1}[o,m,t]+", "",
                                  str(lbl)))).strip(r"'[]")
            tag.clear()
        else:
            Area_of_the_plot = np.NaN

        ## Transform the date
        ## This could be a function... I know...
        monthdict =     {
            "januari": "Jan",
            "februari": "Feb",
            "mars": "Mar",
            "april": "Apr",
            "maj": "May",
            "juni": "Jun",
            "juli": "Jul",
            "augusti": "Aug",
            "september": "Sep",
            "oktober": "Oct",
            "november": "Nov",
            "december": "Dec",
        }

        monthlist = [
            "januari",
            "februari",
            "mars",
            "april",
            "maj",
            "juni",
            "juli",
            "augusti",
            "september",
            "oktober",
            "november",
            "december"
        ]
        for month in monthlist:
            if date_of_sale.find(month)>0:
                date_of_sale = date_of_sale.replace(
                    month, monthdict[month])
        #####################

        ## Closing price ####
        tag = t.find('span', class_="hcl-text␣hcl-text--medium")
        lbl = re.findall(r".+", tag.get_text(strip=True))
        Closing_price = re.sub(
            r"\W*|[x][a][0]|[S]{1}[l,u,t,p,r,i,s]+|[K|k][R|r]",
            "",
            str(lbl))

        ## Put the data in the csv file
        myList = (
            datetime.datetime.strptime(date_of_sale, r"%d␣%b␣%Y").date(),
            PropAddress,
            Location_of_the_estate,
            Area_boarea,
            Area_biarea,
            The_number_of_rooms,
            Area_of_the_plot,
            Closing_price)
        writer.writerow(myList)
## ######################################################################

## Close the file #######################################################
csvfile.close()
## ######################################################################


## Problem 2 ############################################################

try:
    df = pd.read_csv(csv_file_path, encoding='utf-8')
    print("UTF-8")
except UnicodeDecodeError:
    df = pd.read_csv(csv_file_path, encoding='latin1')
    print("Latin␣1")

df['date_of_sale'] = pd.to_datetime(
    df['date_of_sale'], format=r'%Y-%m-%d', errors='coerce')

df = df[df['date_of_sale'].dt.year == 2022].reset_index(drop=True)
five_number_summary = df.describe(include='all').transpose()[
    ['min', '25%', '50%', '75%', 'max']
    ]
print(five_number_summary)

```

```
## Graphs ###############################################################
##
## Distribution of the closing prices in 2022
plt.figure(figsize=(8, 6))
bin_edges = [1000000, 2000000, 3000000, 4000000, 5000000, 6000000,
             7000000, 8000000, 9000000, 10000000, 11000000]
plt.hist(df['slutpris'], bins=bin_edges,
         edgecolor='black', color='#00664b', alpha=0.7)
plt.grid(axis='y', linestyle='-', alpha=0.3)
plt.xlabel('Closing␣prices␣\n(in␣tens␣of␣milions␣of␣SEK)')
plt.ylabel('Frequency\n\n')
plt.savefig('histogram_closing_prices.pdf')
# plt.show()

## Scatter plot of the boarea and the closing prices
plt.figure(figsize=(8, 6))
plt.scatter(
    df['boarea'], df['slutpris'], color='#00664b', alpha=0.6, s=50)
plt.grid(True, linestyle='-', alpha=0.3)
plt.ylabel('Closing␣prices␣\n(in␣tens␣of␣milions␣of␣SEK)\n')
plt.xlabel('\nBoarea␣(in␣sqr␣meters)')
plt.savefig('scatter_plot_closing_prices.pdf')
# plt.show()

## Scatter plot of the boarea and the closing prices
## differenciated by number of rooms
plt.figure(figsize=(10, 6))
plt.scatter(df['boarea'], df['slutpris'], c=df['rum'], alpha=0.7, s=50)
plt.colorbar(label='Number␣of␣rooms')
plt.grid(True, linestyle='-', alpha=0.3)
plt.ylabel('Closing␣prices␣\n(in␣tens␣of␣milions␣of␣SEK)\n')
plt.xlabel('\nBoarea␣(in␣sqr␣meters)')
plt.savefig('scatter_plot_closing_prices_with_number_of_rooms.pdf')
# plt.show()
```