Exploratory data analysis (EDA) of apartments data

Libraries and settings

In [1]:

```
# Libraries
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
import pylab as py

# seaborn graphics settings
sns.set(color_codes=True)

# Ignore warnings
import warnings
warnings.filterwarnings("ignore")

# Show current working directory
print(os.getcwd())
```

```
C:\Users\mmfis\Downloads
```

Univariate non-graphical exploratory data analysis (EDA)

Importing the enriched apartment data

In [2]:

```
# Read and select variables
df_orig = pd.read_csv("apartments_data_enriched.csv")[['web-scraper-order',
                                                       'address_raw',
                                                       'lat',
                                                       'lon',
                                                       'bfs_number',
                                                       'bfs_name',
                                                       'rooms',
                                                       'area',
                                                       'luxurious',
                                                       'price',
                                                       'price_per_m2',
                                                       'pop',
                                                       'pop_dens',
                                                       'emp',
                                                       'frg_pct',
                                                       'mean_taxable_income']]

# Remove duplicates
df_orig = df_orig.drop_duplicates()
df_orig.head(5)

# Remove missing values
```

```python
df_orig = df_orig.dropna()
df_orig.head(5)
```

Out[2]:

	web-scraper-order	address_raw	lat	lon	bfs_number	bfs_name	rooms	area	luxuri
0	1693998201-1	Neuhusstrasse 6, 8630 Rüti ZH, ZH	47.252171	8.845797	118	Rüti (ZH)	3.0	49.0	
1	1693998233-172	Widacherstrasse 5, 8630 Rüti ZH, ZH	47.252087	8.854919	118	Rüti (ZH)	3.0	111.0	
2	1693998256-331	Widenweg 14, 8630 Rüti ZH, ZH	47.253670	8.853993	118	Rüti (ZH)	3.0	58.0	
3	1693998265-381	Rain 1, 8630 Rüti ZH, ZH	47.259834	8.851705	118	Rüti (ZH)	4.0	118.0	
4	1693998276-419	Bachtelstrasse 24b, 8630 Rüti ZH, ZH	47.266113	8.866872	118	Rüti (ZH)	3.0	66.0	

Quantiles original values

In [3]:

```python
df_orig[['price','rooms', 'area', 'price_per_m2', 'pop_dens']].quantile(q=[0.05, 0.
```

Out[3]:

	price	rooms	area	price_per_m2	pop_dens
0.05	1337.00	1.00	26.00	17.90	336.03
0.10	1492.50	1.50	41.50	20.02	525.66
0.25	1842.25	2.50	63.00	23.30	1044.63
0.50	2391.00	3.50	86.00	27.95	1662.60
0.75	3056.75	4.50	108.75	38.12	4778.99
0.90	3960.00	4.75	140.50	52.78	4778.99
0.95	4957.50	5.50	163.75	67.33	4778.99

Filter apartments

In [34]:

```python
df_orig[['area', 'price_per_m2']].describe()
```

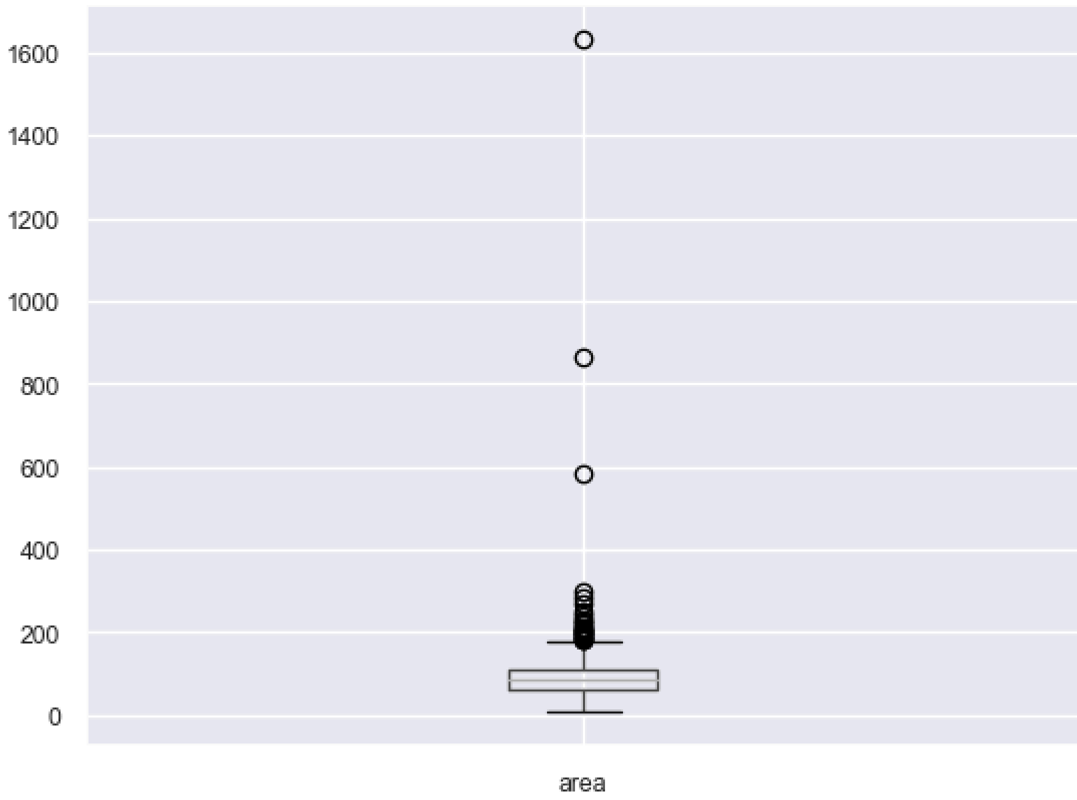
Out[34]:

	area	price_per_m2
count	786.000000	786.000000
mean	92.426209	33.336489
std	75.786527	17.141739
min	8.000000	0.040000
25%	63.000000	23.302500
50%	86.000000	27.950000
75%	108.750000	38.125000
max	1633.000000	149.900000

In [37]:

```
df_orig[['area']].boxplot()
```

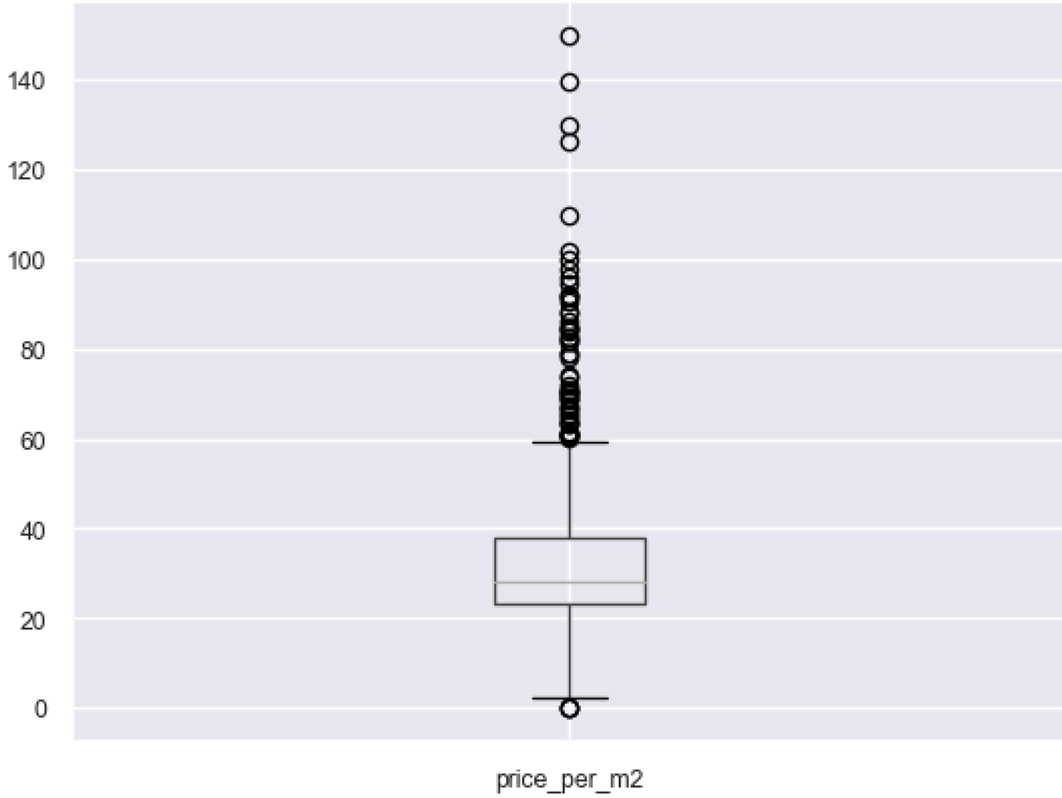
Out[37]: <Axes: >



In [36]:

```
df_orig[['price_per_m2']].boxplot()
```

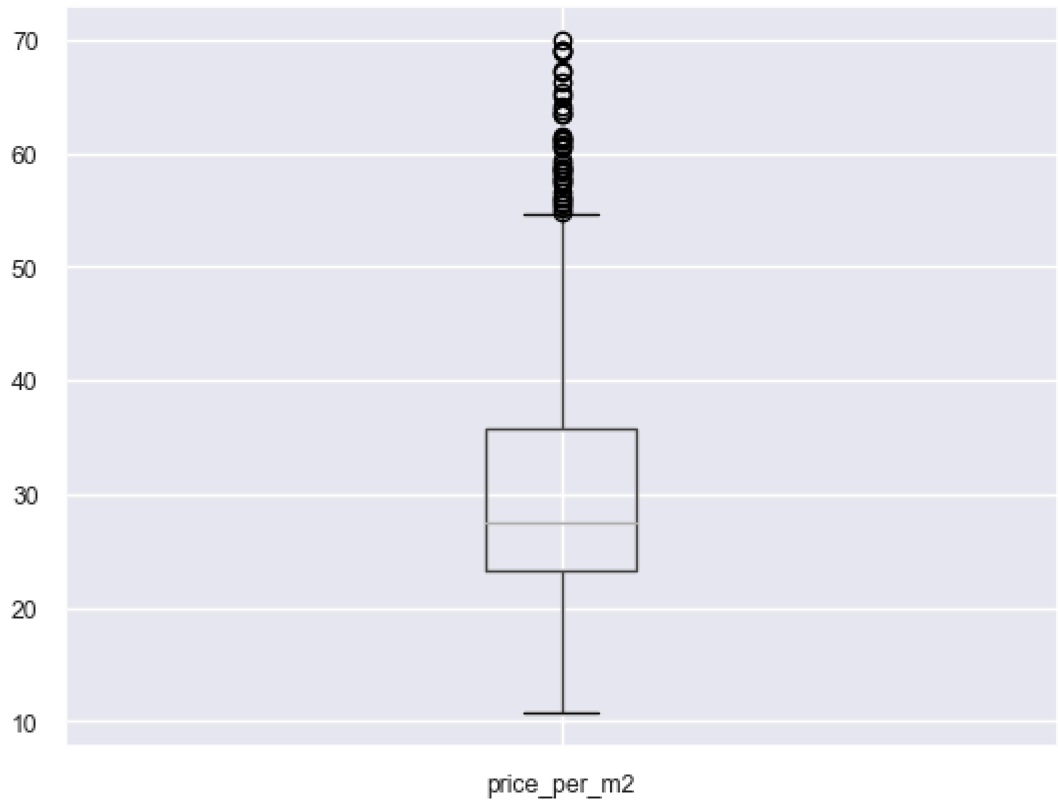
Out[36]: <Axes: >




```
In [77]: # Filter apartments (replace '<= 5000' by the respective operator and quantile)
         df = df_orig.loc[(df_orig['pop_dens'] <= 5000) &
                          (df_orig['area'] <= 200) &
                          (df_orig['price_per_m2'] >= 10) &
                          (df_orig['price_per_m2'] <= 70)]
```

```
In [78]: df[['area']].boxplot()
```

```
Out[78]: <Axes: >
```

In [79]:
```
df[['price_per_m2']].boxplot()
```
Out[79]:
```
<Axes: >
```


Shape (number of rows and colums)

In [80]:
```
# Number of rows and columns
print(df.shape)
```
```
(735, 16)
```

Data types

In [81]:
```
df.dtypes
```
Out[81]:
```
web-scraper-order       object
address_raw             object
lat                    float64
lon                    float64
bfs_number               int64
bfs_name                object
rooms                  float64
area                   float64
luxurious                int64
price                  float64
price_per_m2           float64
pop                      int64
pop_dens               float64
emp                    float64
frg_pct                float64
mean_taxable_income    float64
dtype: object
```

Summary statistics of numeric variables

In [82]:
```
df.describe()
```

Out[82]:

	lat	lon	bfs_number	rooms	area	luxurious	price	price
count	735.000000	735.000000	735.000000	735.000000	735.000000	735.000000	735.000000	73
mean	47.408676	8.603410	183.481633	3.432653	88.609524	0.009524	2569.693878	3
std	0.086464	0.119170	81.650577	1.221356	35.135800	0.097190	1123.881091	1
min	47.195290	8.367652	1.000000	1.000000	8.000000	0.000000	86.000000	1
25%	47.360298	8.517511	118.000000	2.500000	65.000000	0.000000	1837.500000	2
50%	47.398319	8.566746	224.000000	3.500000	86.000000	0.000000	2380.000000	2
75%	47.482353	8.715799	261.000000	4.500000	107.500000	0.000000	2990.000000	3
max	47.693893	8.915933	298.000000	7.500000	200.000000	1.000000	9170.000000	7

Statistical measures (min, max, std, mean, median, count) for selected variables

In [83]:
```
# Price
print('Price:',
      'Count:', round(df.price.count(), 1),
      'Min:', round(df.price.min(), 1),
      'Max:', round(df.price.max(), 1),
      'Mean:', round(df.price.mean(), 1),
      'Median:', round(df.price.median(), 1),
      'Std:', round(df.price.std(), 1))

# Area
print('Area:',
      'Count:', round(df.area.count(), 1),
      'Min:', round(df.area.min(), 1),
      'Max:', round(df.area.max(), 1),
      'Mean:', round(df.area.mean(), 1),
      'Median:', round(df.area.median(), 1),
      'Std:', round(df.area.std(), 1))
```

```
Price: Count: 735 Min: 86.0 Max: 9170.0 Mean: 2569.7 Median: 2380.0 Std: 1123.9
Area: Count: 735 Min: 8.0 Max: 200.0 Mean: 88.6 Median: 86.0 Std: 35.1
```

Skewness

In [84]:
```
df[['price','rooms', 'area']].skew()
```

Out[84]:
```
price     1.981442
rooms     0.030231
area      0.443609
dtype: float64
```

Kurtosis

In [85]:
```
df[['price','rooms', 'area']].kurtosis()
```

Out[85]:
```
price     6.752420
rooms    -0.086465
area      0.266880
dtype: float64
```

Extreme values

In [86]:
```
# Low costs apartments
df[df['price_per_m2'] <= 10]
```

Out[86]:

web-scraper-order	address_raw	lat	lon	bfs_number	bfs_name	rooms	area	luxurious	price	price_per

In [87]:
```
# Very expansive apartments
df[df['price_per_m2'] >= 100]
```

Out[87]:

web-scraper-order	address_raw	lat	lon	bfs_number	bfs_name	rooms	area	luxurious	price	price_per

Get a list of categories of categorical variable

In [88]:
```
np.array(pd.Categorical(df['bfs_name']).categories)
```

Out[88]:
```
array(['Adliswil', 'Aeugst am Albis', 'Affoltern am Albis', 'Altikon',
       'Andelfingen', 'Bachenbülach', 'Bassersdorf', 'Bauma',
       'Bonstetten', 'Bülach', 'Dielsdorf', 'Dietikon', 'Dietlikon',
       'Dättlikon', 'Dübendorf', 'Dürnten', 'Egg', 'Eglisau', 'Elsau',
       'Embrach', 'Fehraltorf', 'Feuerthalen', 'Freienstein-Teufen',
       'Fällanden', 'Glattfelden', 'Gossau (ZH)', 'Greifensee',
       'Hausen am Albis', 'Hedingen', 'Herrliberg', 'Hettlingen',
       'Hinwil', 'Hittnau', 'Hochfelden', 'Hombrechtikon', 'Höri',
       'Hüttikon', 'Kloten', 'Knonau', 'Küsnacht (ZH)',
       'Langnau am Albis', 'Laufen-Uhwiesen', 'Lindau', 'Lufingen',
       'Maur', 'Meilen', 'Mettmenstetten', 'Männedorf', 'Mönchaltorf',
       'Neerach', 'Neftenbach', 'Niederglatt', 'Niederhasli',
       'Niederweningen', 'Nürensdorf', 'Oberengstringen', 'Oberglatt',
       'Obfelden', 'Oetwil am See', 'Oetwil an der Limmat', 'Opfikon',
       'Ossingen', 'Pfungen', 'Pfäffikon', 'Regensdorf', 'Rheinau',
       'Richterswil', 'Rickenbach (ZH)', 'Rorbas', 'Russikon', 'Rümlang',
       'Rüschlikon', 'Rüti (ZH)', 'Schlatt (ZH)', 'Schlieren',
       'Schwerzenbach', 'Seuzach', 'Stadel', 'Stallikon', 'Steinmaur',
       'Stäfa', 'Thalwil', 'Trüllikon', 'Uitikon', 'Urdorf', 'Uster',
       'Volketswil', 'Wald (ZH)', 'Wallisellen', 'Wangen-Brüttisellen',
       'Weiach', 'Weiningen (ZH)', 'Wettswil am Albis', 'Wetzikon (ZH)',
       'Wiesendangen', 'Wila', 'Winkel', 'Winterthur', 'Zell (ZH)',
       'Zollikon', 'Zürich'], dtype=object)
```

Multivariate non-graphical exploratory data analysis (EDA)

Cross-tabulation

In [89]:
```
pd.crosstab(df['luxurious'], df['rooms'])
```

Out[89]:

rooms	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0	6.5	7.0	7.5
luxurious														
0	46	23	34	128	44	200	27	161	3	49	1	9	2	1
1	0	0	0	5	0	2	0	0	0	0	0	0	0	0

Pivot tables

In [90]:
```
# Using pivot_table to reshape the data and calculate means
pd.pivot_table(df[['rooms', 'price', 'price_per_m2', 'area', 'luxurious']],
               index=['rooms', 'luxurious'],
               values=['price', 'price_per_m2', 'area'],
               aggfunc=(np.mean, 'count'))
```

Out[90]:

		area		price		price_per_m2	
		count	mean	count	mean	count	mean
rooms	**luxurious**						
1.0	**0**	46	34.239130	46	1288.652174	46	43.765652
1.5	**0**	23	40.521739	23	1919.826087	23	49.796957
2.0	**0**	34	56.735294	34	1926.058824	34	34.855588
2.5	**0**	128	67.250000	128	2245.953125	128	34.584375
	1	5	71.600000	5	2627.400000	5	36.774000
3.0	**0**	44	68.159091	44	1964.022727	44	28.677955
3.5	**0**	200	89.770000	200	2648.980000	200	29.494700
	1	2	102.500000	2	4480.000000	2	43.990000
4.0	**0**	27	93.222222	27	2828.370370	27	30.204444
4.5	**0**	161	114.770186	161	3010.378882	161	25.922547
5.0	**0**	3	108.000000	3	2881.666667	3	26.376667
5.5	**0**	49	146.632653	49	3420.693878	49	23.501837
6.0	**0**	1	150.000000	1	4800.000000	1	32.000000
6.5	**0**	9	151.555556	9	3835.222222	9	25.181111
7.0	**0**	2	175.000000	2	6350.000000	2	36.325000
7.5	**0**	1	200.000000	1	4700.000000	1	23.500000

Correlation matrix

In [91]:
```
corr = df[['rooms',
          'area',
          'price',
          'price_per_m2',
          'pop_dens',
```

```
              'frg_pct']].cov().corr()
corr
```

Out[91]:

	rooms	area	price	price_per_m2	pop_dens	frg_pct
rooms	1.000000	0.979639	0.459467	-0.541385	-0.616754	-0.690699
area	0.979639	1.000000	0.628409	-0.361585	-0.446165	-0.531460
price	0.459467	0.628409	1.000000	0.498022	0.415646	0.324792
price_per_m2	-0.541385	-0.361585	0.498022	1.000000	0.995599	0.981784
pop_dens	-0.616754	-0.446165	0.415646	0.995599	1.000000	0.995208
frg_pct	-0.690699	-0.531460	0.324792	0.981784	0.995208	1.000000

Covariance matrix

In [92]:

```
cov = df[['rooms',
          'area',
          'price',
          'price_per_m2',
          'pop_dens',
          'frg_pct']].cov()
cov
```

Out[92]:

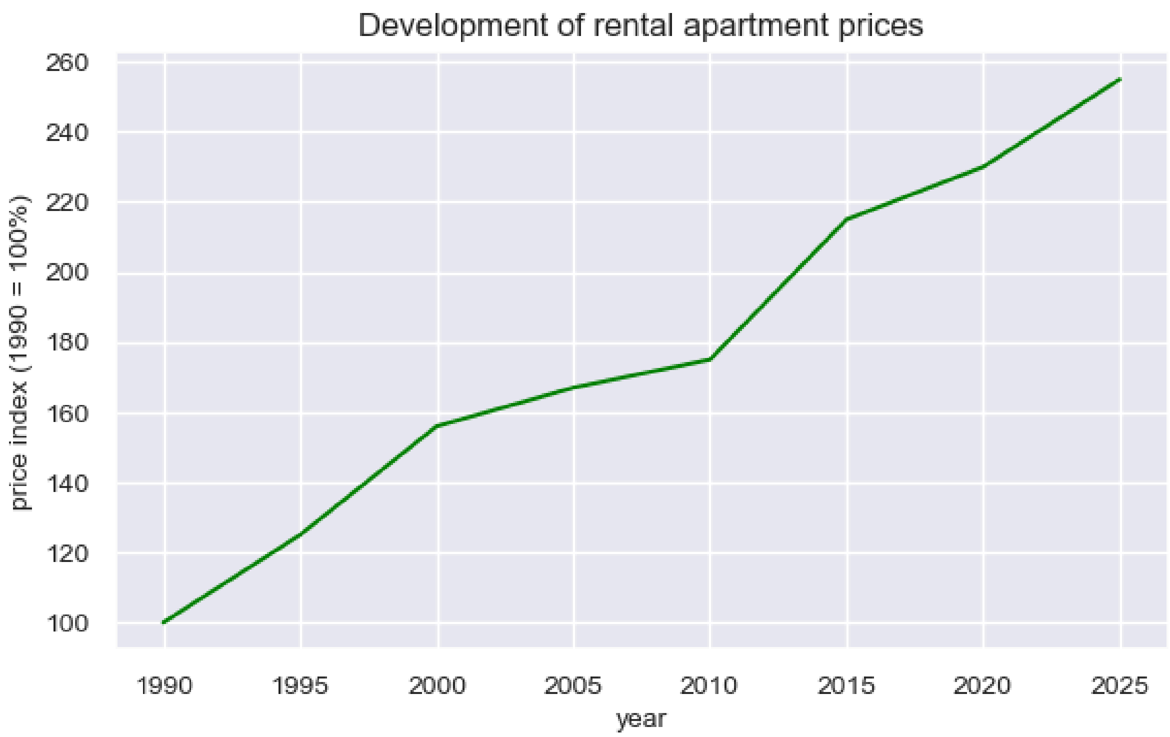
	rooms	area	price	price_per_m2	pop_dens	frg_pct
rooms	1.491712	35.844920	6.791374e+02	-6.399997	-6.167334e+02	-2.315958
area	35.844920	1234.524432	2.599311e+04	-176.543591	-1.390285e+04	-58.691324
price	679.137394	25993.113274	1.263109e+06	3396.661496	5.570847e+05	790.341413
price_per_m2	-6.399997	-176.543591	3.396661e+03	123.370531	1.210049e+04	30.775488
pop_dens	-616.733400	-13902.847608	5.570847e+05	12100.490956	2.928563e+06	8381.117044
frg_pct	-2.315958	-58.691324	7.903414e+02	30.775488	8.381117e+03	57.482422

Univariate graphical exploratory data analysis (EDA)

Line chart (matplotlib)

In [93]:

```
# Generate some useful values (time series)
x = [1990, 1995, 2000, 2005, 2010, 2015, 2020, 2025]
y = [100, 125, 156, 167, 175, 215, 230, 255]

# Create figure
fig = plt.figure(figsize=(7,4))
plt.plot(x, y, color="green")
plt.title('Development of rental apartment prices', fontsize=12)
plt.xlabel('year', fontsize=10)
plt.ylabel('price index (1990 = 100%)', fontsize=10)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)

plt.show()
```



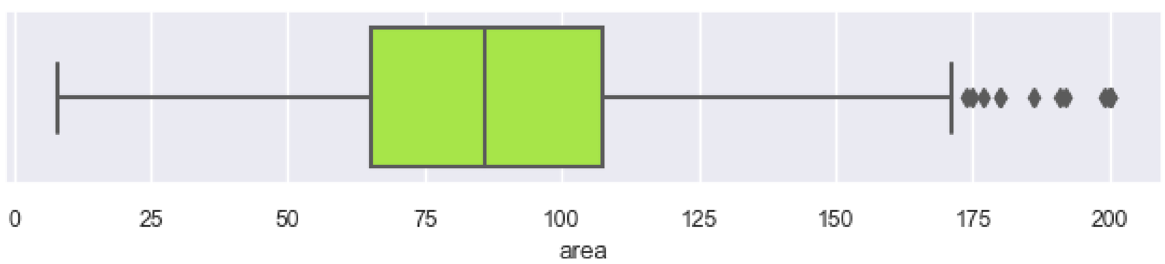
Boxplot (seaborn)

In [94]:

```
plt.figure(figsize=(8,1.2))
plt.ticklabel_format(style='plain')
sns.boxplot(x=df['area'], color="greenyellow")
```

Out[94]:

```
<Axes: xlabel='area'>
```

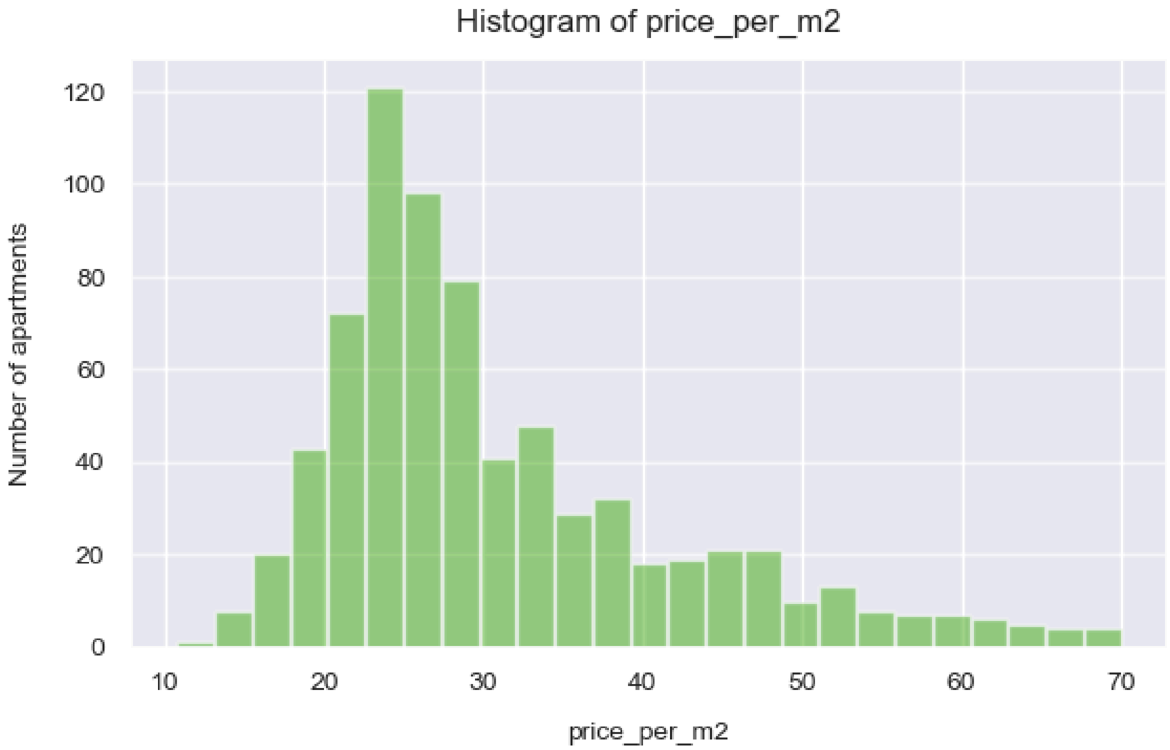


Histogram (matplotlib)

In [95]:

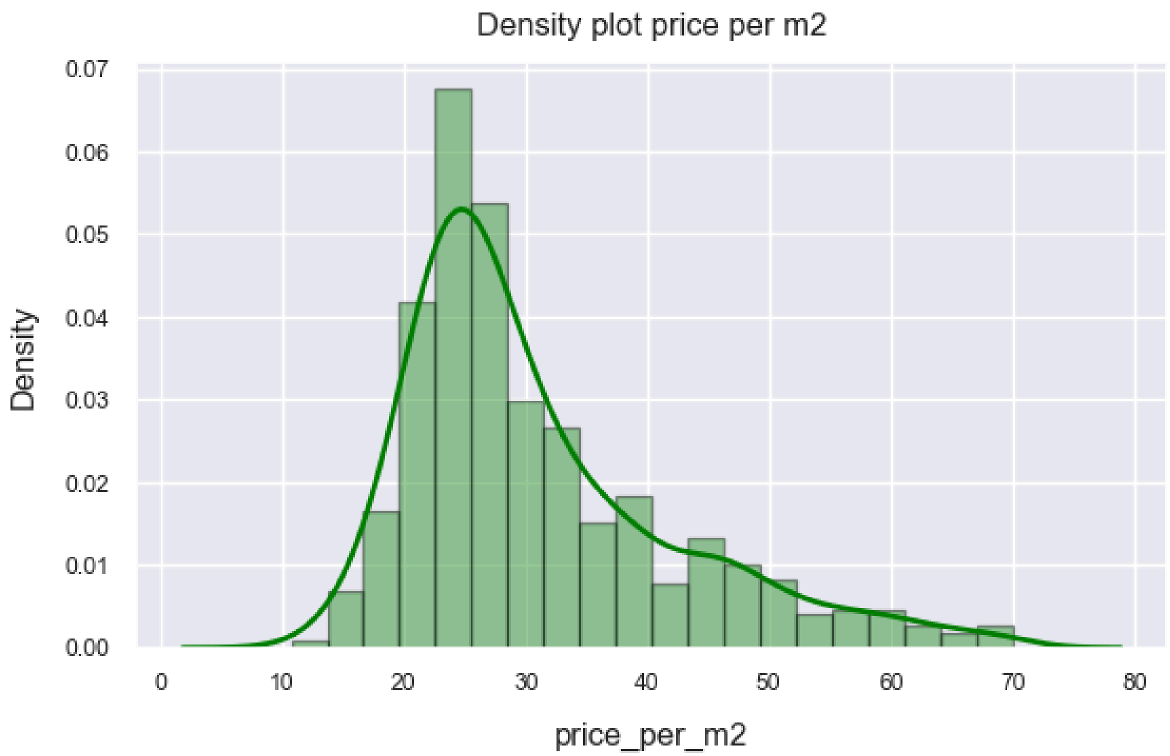
```
import matplotlib.pyplot as plt
import numpy as np

# Plot Histogram
fig = plt.figure( figsize=(7,4))

plt.xticks(fontsize=14, rotation=0)
plt.yticks(fontsize=14, rotation=0)

n, bins, patches = plt.hist(x=df['price_per_m2'],
                            bins=25,
                            color='#42AD12',
                            alpha=0.5,
                            rwidth=0.95
                   )
plt.grid(True)
plt.ticklabel_format(style='plain')
plt.grid(axis='y', alpha=0.75)
```

```python
# Set labels
plt.xlabel('price_per_m2', fontsize=10, labelpad=10)
plt.ylabel('Number of apartments', fontsize=10, labelpad=10)
plt.title('Histogram of price_per_m2', fontsize=12, pad=10)

# Set fontsize of tick labels
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10)

plt.show()
```



Density plot (seaborn)

In [96]:
```python
plt.figure(figsize=(7,4))
sns.distplot(df['price_per_m2'],
             hist=True,
             kde=True,
             bins=20,
             color = 'green',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 2},
            )
plt.title('Density plot price per m2', fontsize=12, pad=10)
plt.xlabel('price_per_m2', fontsize=12, labelpad=10)
plt.ylabel('Density', fontsize=12, labelpad=10)
plt.grid(True)
plt.show()
```

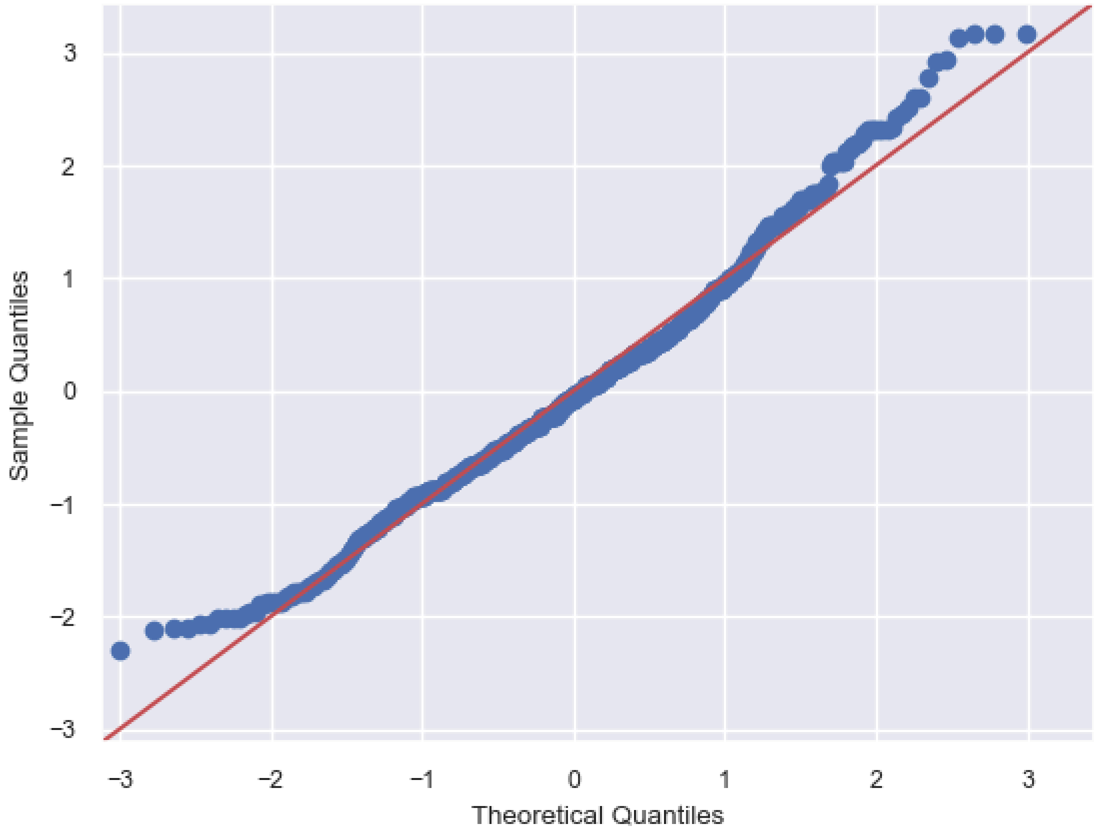


Quantile-Quantile (QQ-) plot

In [97]:

```
# Variable (standardized)
x = 'area'
df_qq = df
df_qq['var'] = (df[x]-df[x].mean()) / df[x].std()
print(df_qq.sort_values('var')[['area', 'var']])

# Plot
sm.qqplot(df_qq['var'], line ='45')
py.show()
```

```
      area       var
682    8.0 -2.294228
190   14.0 -2.123462
431   15.0 -2.095001
52    15.0 -2.095001
430   16.0 -2.066540
..     ...       ...
13   192.0  2.942596
552  199.0  3.141823
320  200.0  3.170284
777  200.0  3.170284
733  200.0  3.170284

[735 rows x 2 columns]
```

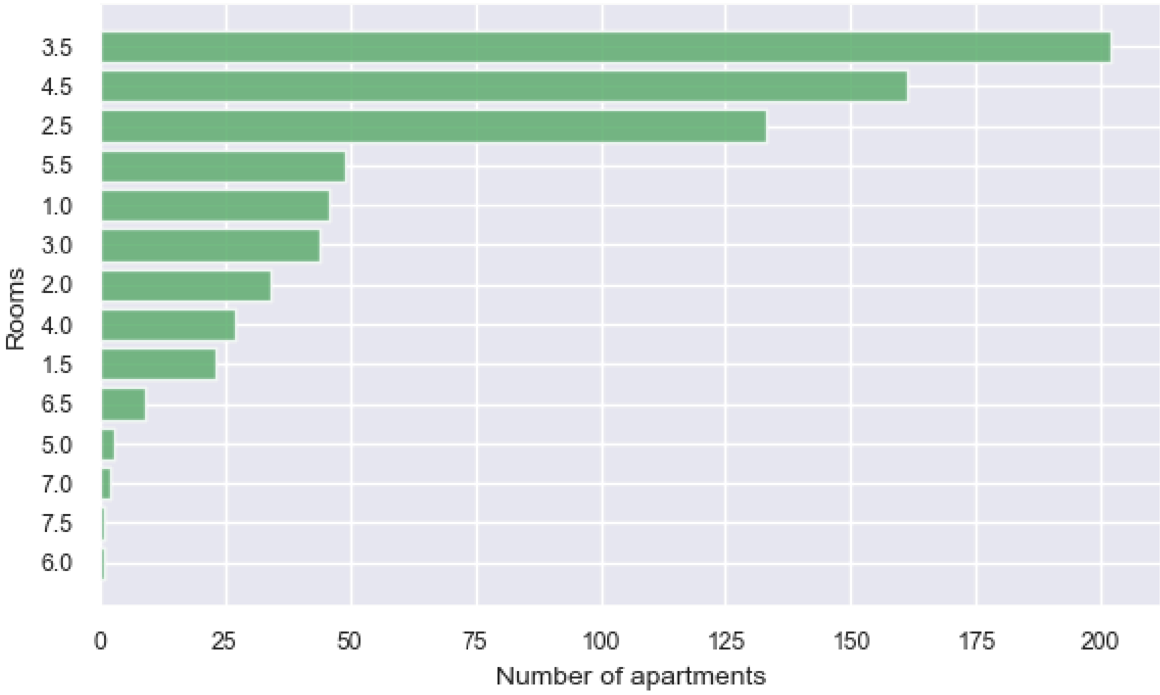


Barchart (matplotlib)

In [98]:
```
# Group data by rooms (only the topmost 15 values are shown)
df_bar = df['rooms'].value_counts().nlargest(15).sort_values(ascending=True)

# Values for barchart
napart = list(df_bar.values)
index = list(df_bar.index.values)
index
```

Out[98]:
```
[6.0, 7.5, 7.0, 5.0, 6.5, 1.5, 4.0, 2.0, 3.0, 1.0, 5.5, 2.5, 4.5, 3.5]
```

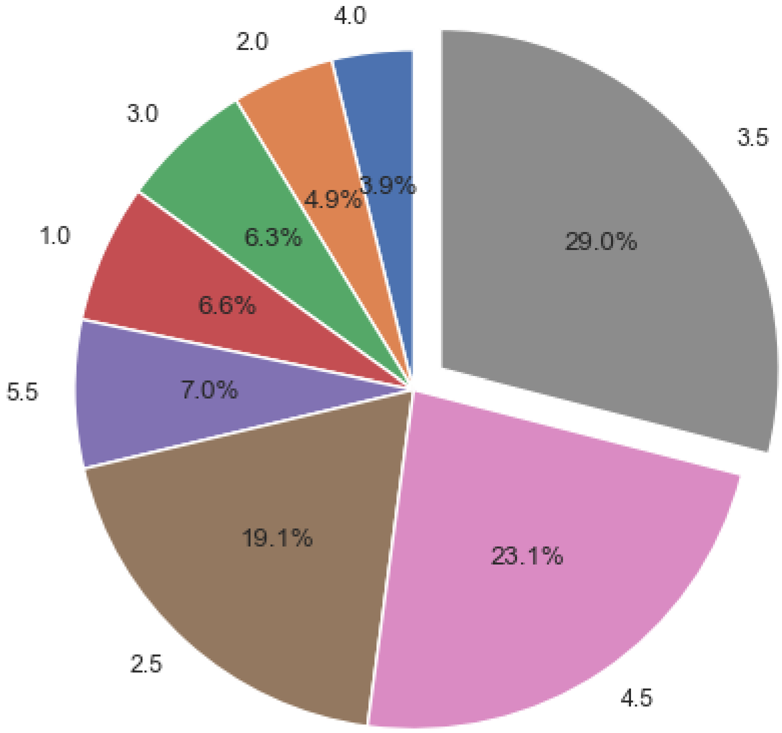
In [99]:
```
# Group data by rooms (only the topmost 15 values are shown)
df_bar = df['rooms'].value_counts().nlargest(15).sort_values(ascending=True)

# Values for barchart
napart = list(df_bar.values)
index = list(df_bar.index.values)
y_pos = np.arange(len(index))

# Figure
fig, ax = plt.subplots(figsize=(7,4))
ax.barh(y_pos, napart, align='center', color='g', alpha=0.8)
ax.set_yticks(y_pos, index)
ax.set_xlabel('Number of apartments', fontsize=10)
ax.set_ylabel('Rooms', fontsize=10)

# Show graph
plt.show()
```



Piechart (matplotlib)

In [100…

```
# Group data by rooms (only the 8 most frequently occurencies by rooms)
df_bar = df.rooms.value_counts().nlargest(8).sort_values(ascending=True)

# Simple bar chart
sizes = list(df_bar.values)
labels = list(df_bar.index.values)
explode = (0, 0, 0, 0, 0.0, 0, 0, 0.1) # increases distance of pieces

fig1, ax1 = plt.subplots(figsize=(5,5))
ax1.pie(sizes,
        labels=labels,
        explode=explode,
        autopct='%1.1f%%',
        shadow=False,
        startangle=90)
ax1.axis('equal')  # ensures that pie is drawn as a circle.

plt.show()
```

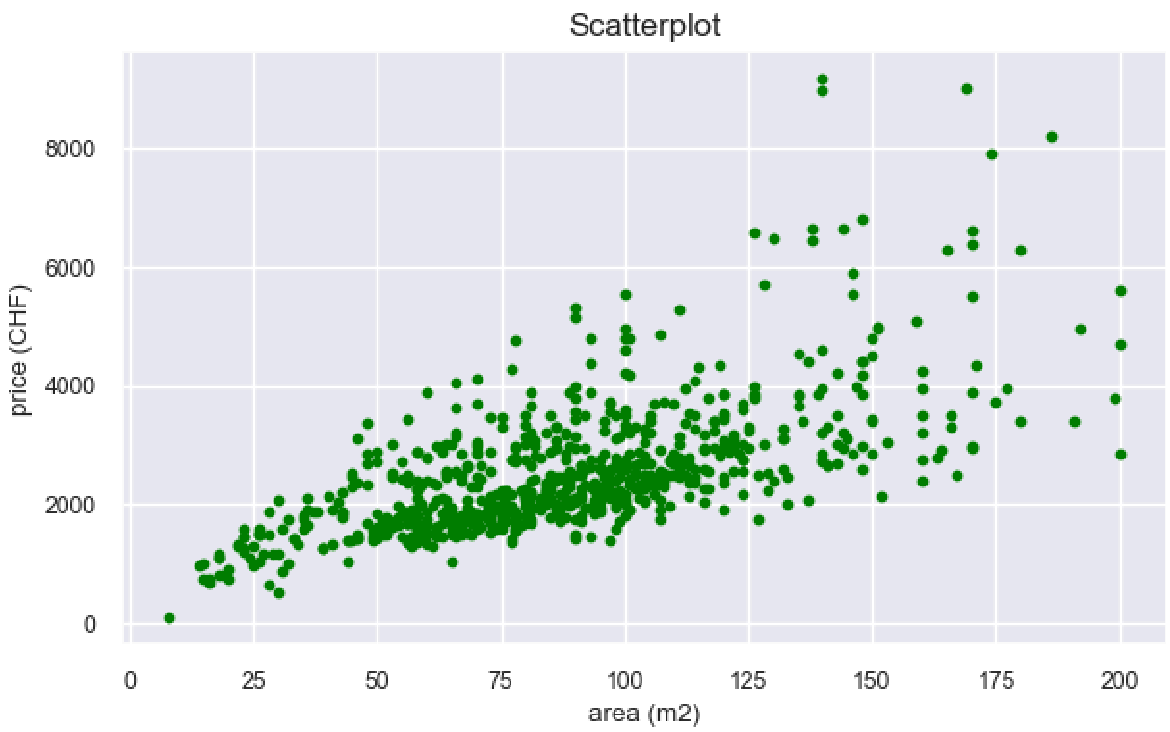


Multivariate graphical exploratory data analysis (EDA)

Scatterplot (matplotlib)

In [101…

```
plt.figure(figsize=(7,4))
plt.scatter(df['area'],
            df['price'],
            color="green",
            alpha=1.0,
            s=10)
plt.title('Scatterplot', fontsize=12)
plt.xlabel('area (m2)')
plt.ylabel('price (CHF)')
plt.show()
```

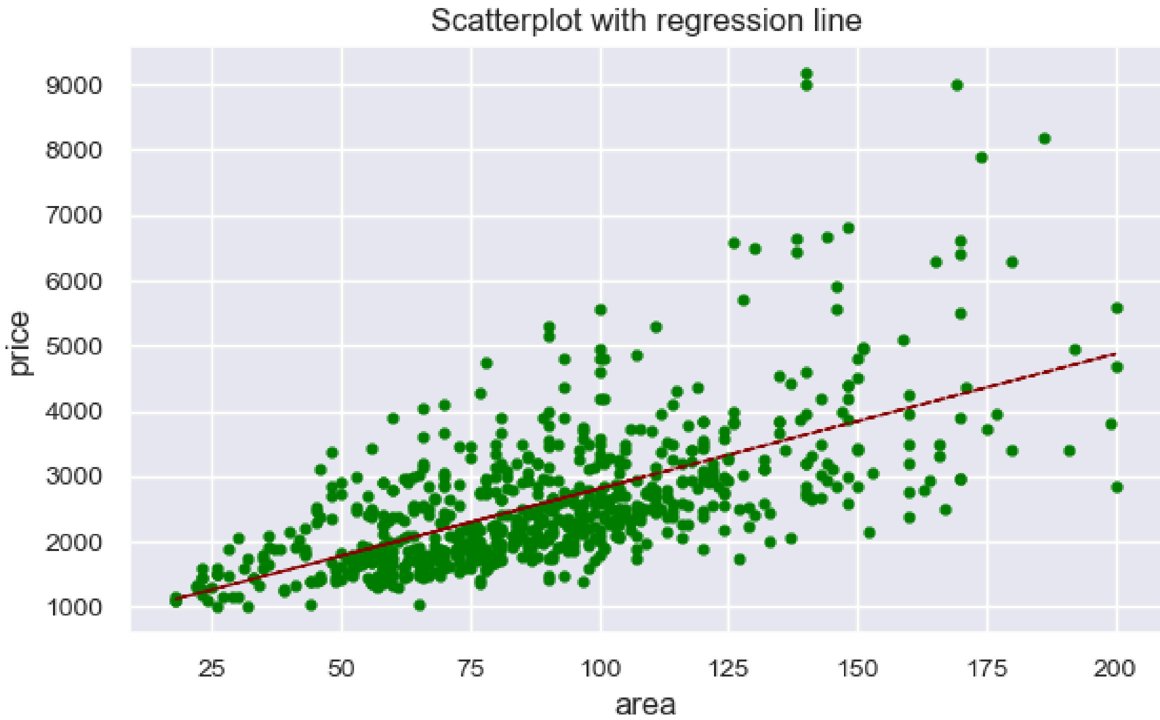


Scatterplot (matplotlib) with regression line

In [102…

```
# Subset
df_sub = df.loc[(df.price >= 1000)]
print(df_sub.shape)

# Scatterplot
plt.figure(figsize=(7,4))
plt.plot(df_sub.area,
         df_sub.price,
         'o',
         markersize=3.5,
         color="green")

# Regression line (b = slope, a=intercept)
b, a = np.polyfit(df_sub.area, df_sub.price, 1)
print(b)
print(a)

# Plot regression line
plt.plot(df_sub.area,
         b*df_sub.area + a,
         linewidth=1,
         linestyle='dashed',
         color='darkred')

# Add title and axes labels
plt.title('Scatterplot with regression line', fontsize=12)
plt.ylabel('price', fontsize=12)
plt.xlabel('area', fontsize=12)

# Set fontsize of tick labels
plt.xticks(fontsize = 10)
plt.yticks(fontsize = 10)

plt.show()
```

```
(721, 17)
20.63221634682636
749.185677197324
```



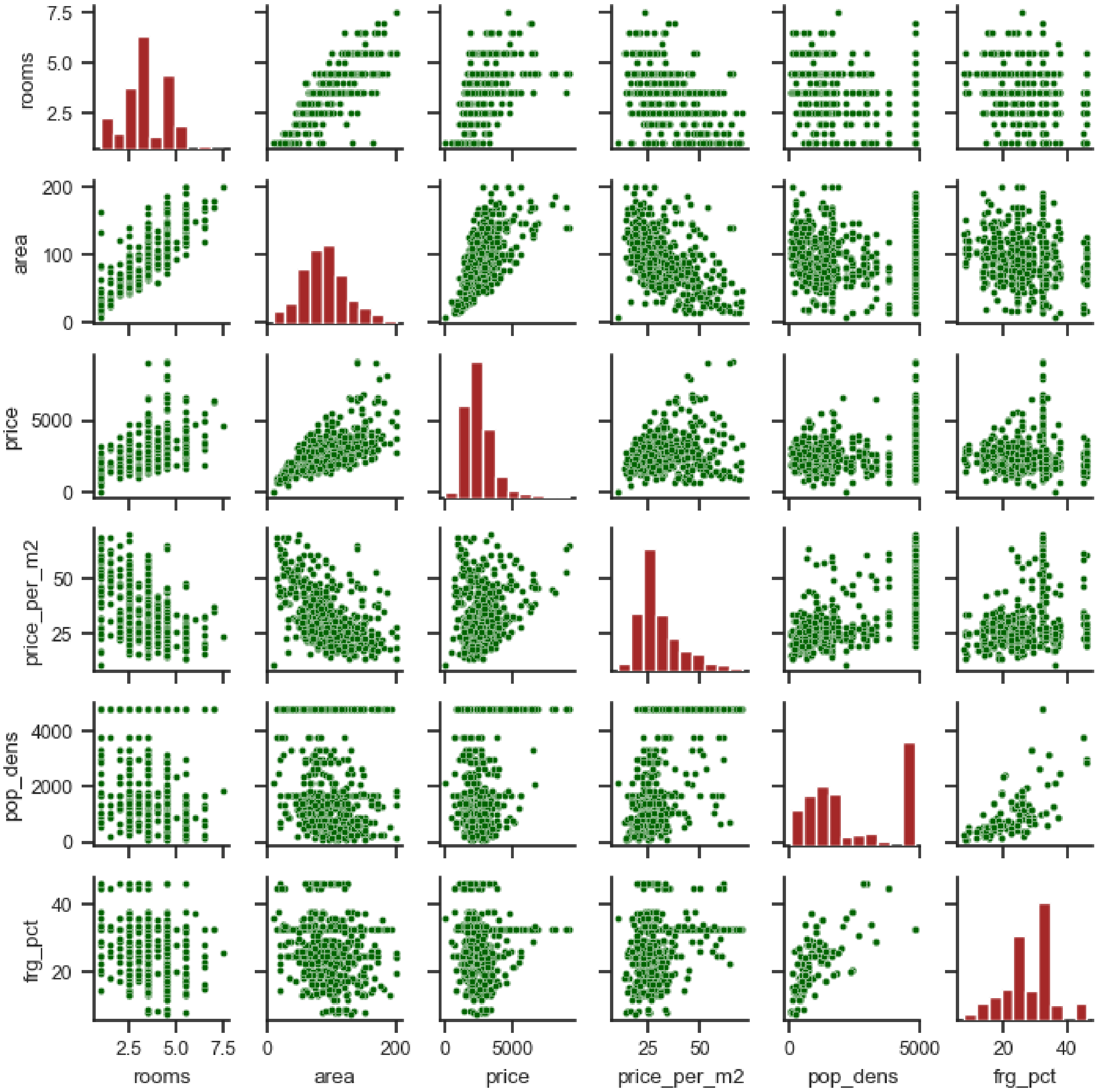
Scatterplot-matrix (seaborn)

In [103…

```
sns.set(style="ticks", font_scale=0.8)
g = sns.PairGrid(df[['rooms',
                     'area',
                     'price',
                     'price_per_m2',
                     'pop_dens',
                     'frg_pct']],
                 height=1.2,
                 aspect=1)
g.map_upper(sns.scatterplot,color='darkgreen', s=10)
g.map_lower(sns.scatterplot, color='darkgreen', s=10)
g.map_diag(plt.hist, color='brown')
```

Out[103]:

```
<seaborn.axisgrid.PairGrid at 0x1d03ab3c710>
```

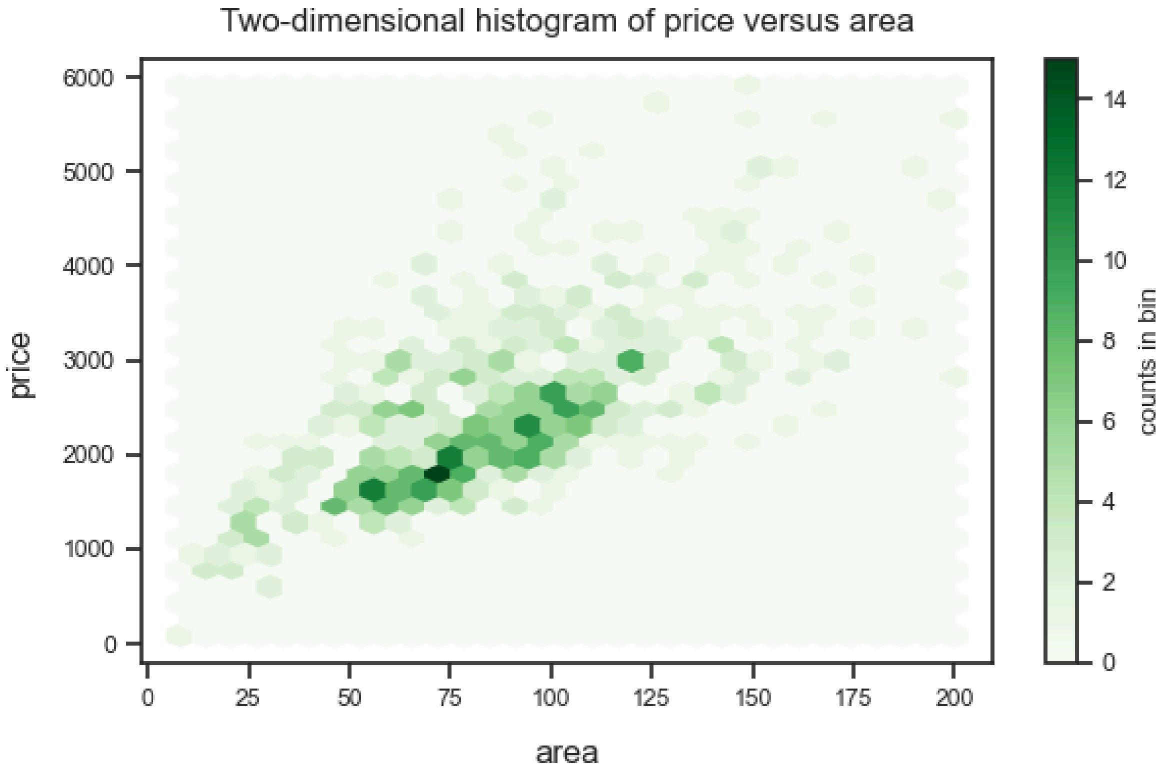


Hexagonal binning plot (matplotlib)

In [104…

```
# Subset
df_sub = df.loc[(df.price <= 6000) & (df.area <= 200)]
print(df_sub.shape)

# Plot
fig = plt.figure( figsize=(7,4) )
plt.hexbin(df_sub.area, df_sub.price, gridsize=30, cmap='Greens')

# Set labels
plt.xlabel('area', fontsize=12, labelpad=10)
plt.ylabel('price', fontsize=12, labelpad=10)
plt.title('Two-dimensional histogram of price versus area', fontsize=12, pad=10)

cb = plt.colorbar(label='count in bin')
cb.set_label('counts in bin')
```

```
(720, 17)
```



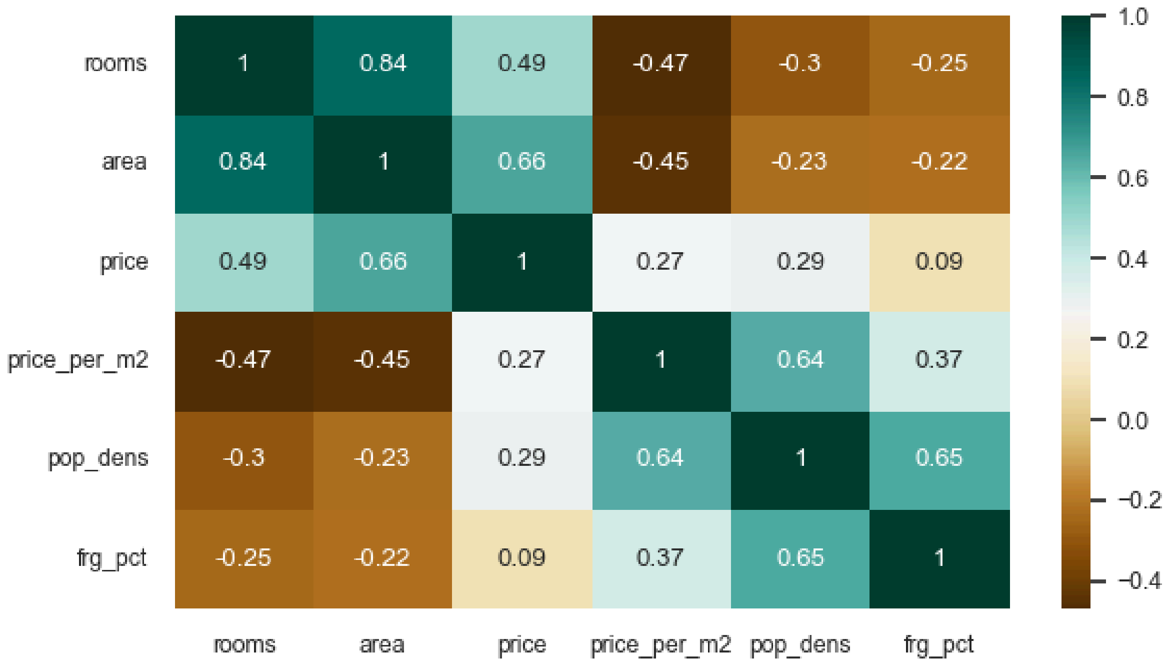
Correlation heatmap (seaborn)

In [105…

```
# Set font size for plot
sns.set(font_scale=0.8)

# Create correlation matrix as the basis for the heatmap
plt.figure(figsize=(7,4))
corr = df[['rooms',
                     'area',
                     'price',
                     'price_per_m2',
                     'pop_dens',
                     'frg_pct']].corr().round(2)

# Plot heatmap
sns.heatmap(corr,
            cmap="BrBG",
            annot=True)
```

Out[105]:

```
<Axes: >
```



Bubble plot (seaborn)

In [106…

```
# Subset of df
df_sub = df.loc[(df['rooms'] >= 2.5) & (df['rooms'] <= 4.5)]

plt.figure(figsize=(7,4))
plt.ticklabel_format(style='plain')
cmap = sns.cubehelix_palette(dark=.3, light=3, as_cmap=True)

ax = sns.scatterplot(x="area",
                     y="price",
                     size="rooms", # determines bubble size
                     hue="pop_dens",  # determines color
                     palette="Set2",
                     data=df_sub)

# Set title and axes
ax.set_title('Price vs area', fontsize = 12)
ax.set_xlabel('area', fontsize = 10)
ax.set_ylabel('price', fontsize = 10)
ax.legend([],[], frameon=False) # skip legend
```

Out[106]:

```
<matplotlib.legend.Legend at 0x1d03f3eb810>
```

Jupyter notebook --footer info-- (please always provide this at the end of each submitted notebook)

In [107…

```
import os
import platform
import socket
from platform import python_version
from datetime import datetime

print('-----------------------------------')
print(os.name.upper())
print(platform.system(), '|', platform.release())
print('Datetime:', datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
print('Python Version:', python_version())
print('-----------------------------------')
```

```
-----------------------------------
NT
Windows | 10
Datetime: 2024-10-08 15:28:44
Python Version: 3.11.5
-----------------------------------
```