

eda

August 6, 2025

1 flats-in-cracow exploratory data analysis

1.1 Imports

```
[3]: import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

from sklearn.impute import KNNImputer
from pylab import rcParams
from pathlib import Path
```

1.2 Setup

```
[4]: # Create directory for images
Path("img").mkdir(parents=True, exist_ok=True)

# Set default figure size
rcParams['figure.figsize'] = (4, 4)

# Tell pandas how to display floats
pd.options.display.float_format = "{:,.2f}".format
```

1.3 Data loading

```
[5]: data = pd.read_csv('cleaned_real_estate.csv')
```

```
[6]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1767 entries, 0 to 1766
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   District    1767 non-null   object
 1   Amount      1767 non-null   int64
 2   Seller      1767 non-null   object
```

```

3   Area      1767 non-null  int64
4   Rooms     1767 non-null  int64
5   Bathrooms 1767 non-null  int64
6   Parking   1767 non-null  object
7   Garden    1767 non-null  bool
8   Balcony   1767 non-null  bool
9   Terrace   1767 non-null  bool
10  Floor     1767 non-null  bool
11  New       1767 non-null  bool
12  Estate    1767 non-null  bool
13  Townhouse 1767 non-null  bool
14  Apartment 1767 non-null  bool
15  Land      1767 non-null  bool
16  Studio    1767 non-null  bool
dtypes: bool(10), int64(4), object(3)
memory usage: 114.0+ KB

```

```
[7]: data.head()
```

```

[7]:   District  Amount  Seller  Area  Rooms  Bathrooms  Parking  Garden \
0   biezanow  536505  realtor   22     1           1 no parking  False
1   bieczyce  646975  realtor   46     2           1 no parking  True
2   lagiewniki 816233  realtor   37     1           1 no parking  False
3   zwierzyniec 1009826  realtor   55     2           2 no parking  True
4   bronowice  733546  realtor   67     2           1   garage  True

   Balcony  Terrace  Floor  New  Estate  Townhouse  Apartment  Land  Studio
0   False   False  False  False  False   False   False   False  False
1    True   False  False  False  False   True   False   False  False
2   False   True  False  True  False   False   False   False  False
3   False  False  False  True  False   True   False   False  False
4   False  False  False  True  False   False   False   False  False

```

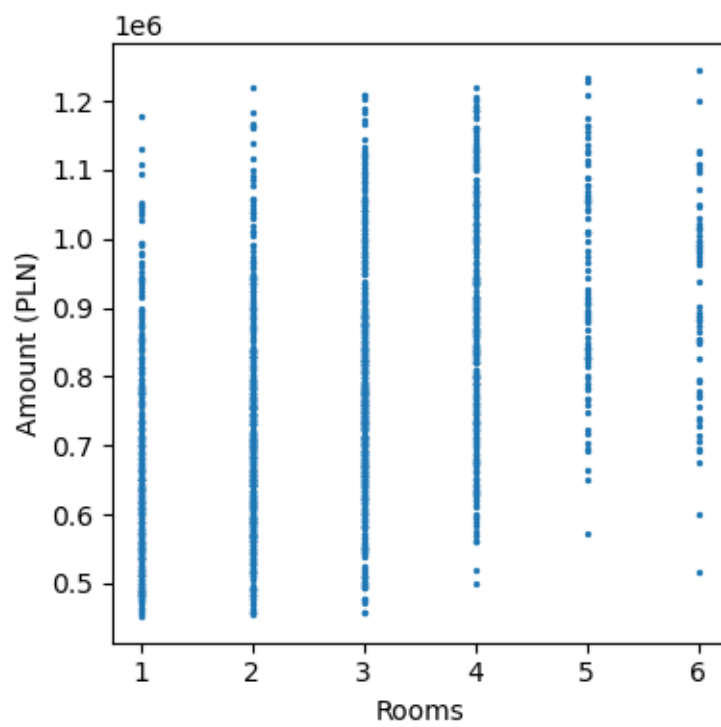
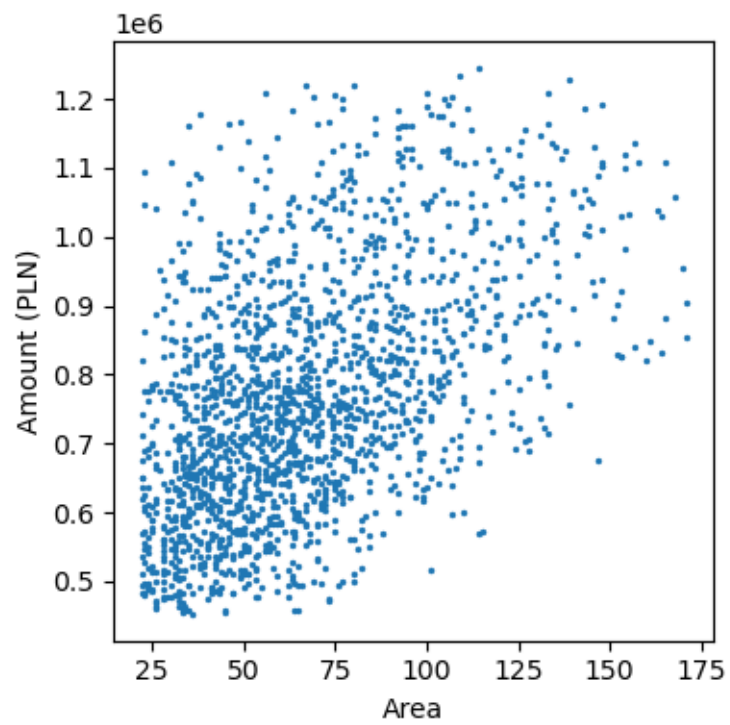
1.4 Numeric features

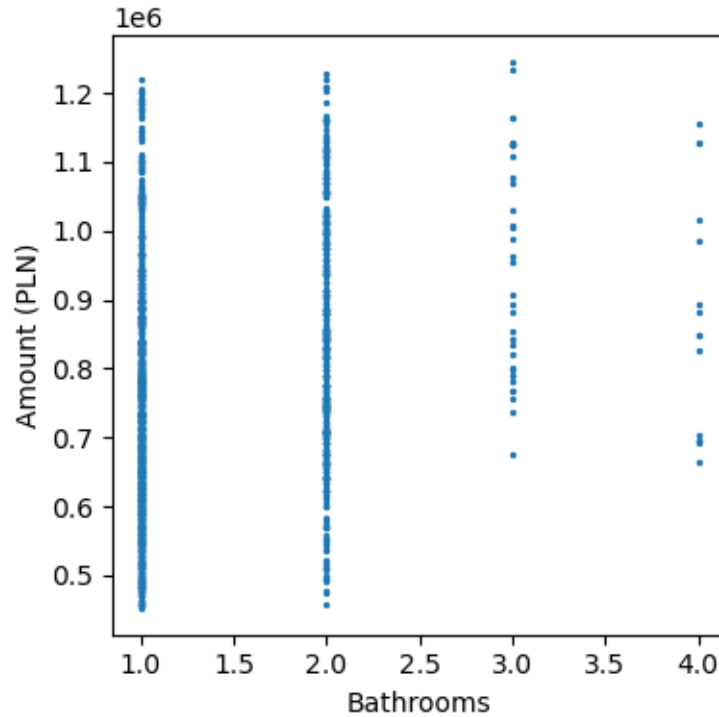
To visually inspect the data we are going to make scatterplot for each of the numeric columns.

```

[8]: numeric = list(data.select_dtypes('number').columns)
for col in numeric:
    if col != 'Amount':
        plt.scatter(data[col], data['Amount'], s=2)
        plt.xlabel(f'{col}')
        plt.ylabel(f'Amount (PLN)')
        plt.tight_layout()
        plt.show()

```





```
[9]: data.select_dtypes('number').corr()
```

```
[9]:
```

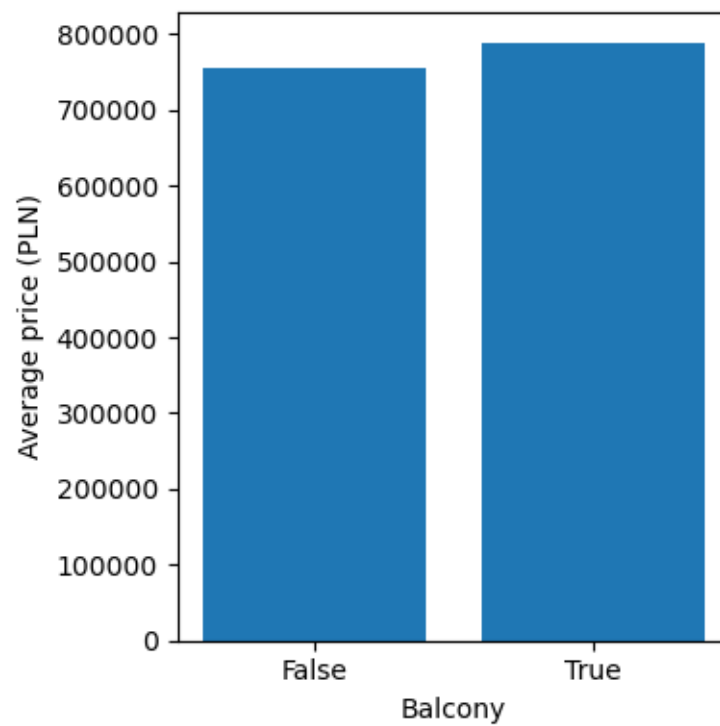
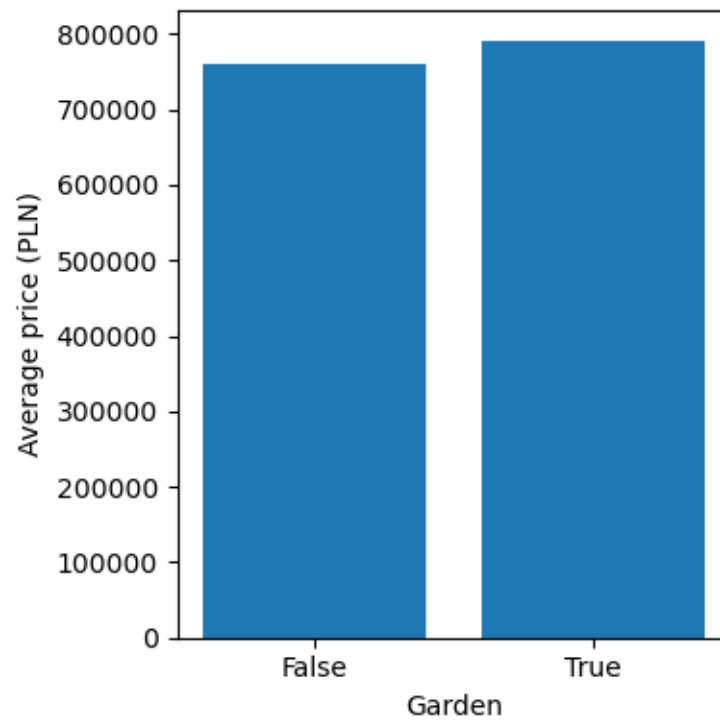
	Amount	Area	Rooms	Bathrooms
Amount	1.00	0.51	0.45	0.31
Area	0.51	1.00	0.86	0.37
Rooms	0.45	0.86	1.00	0.43
Bathrooms	0.31	0.37	0.43	1.00

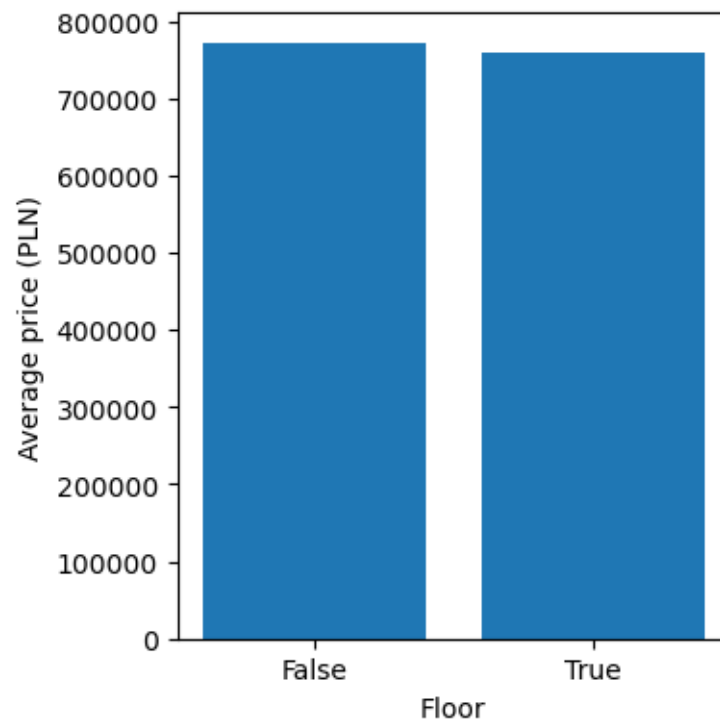
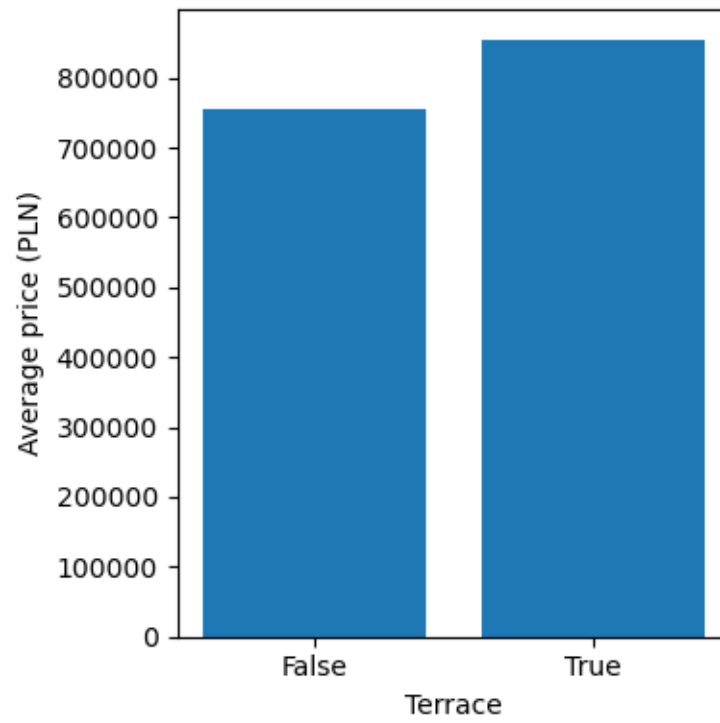
1.5 Binary features

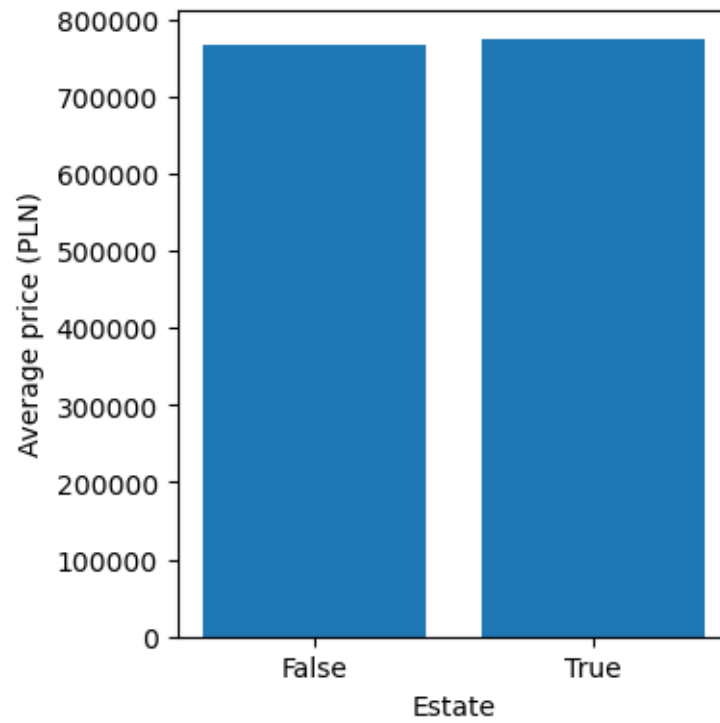
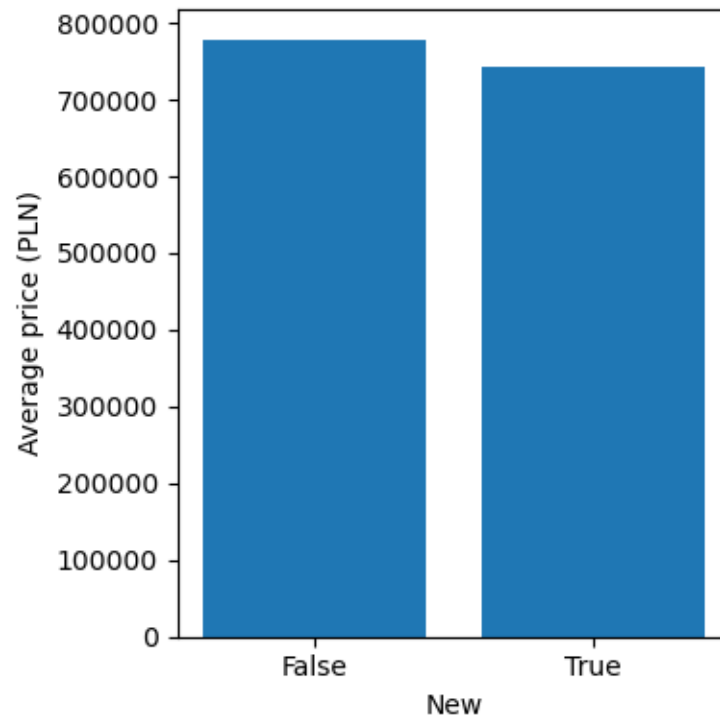
We are going to group the data and compare averages.

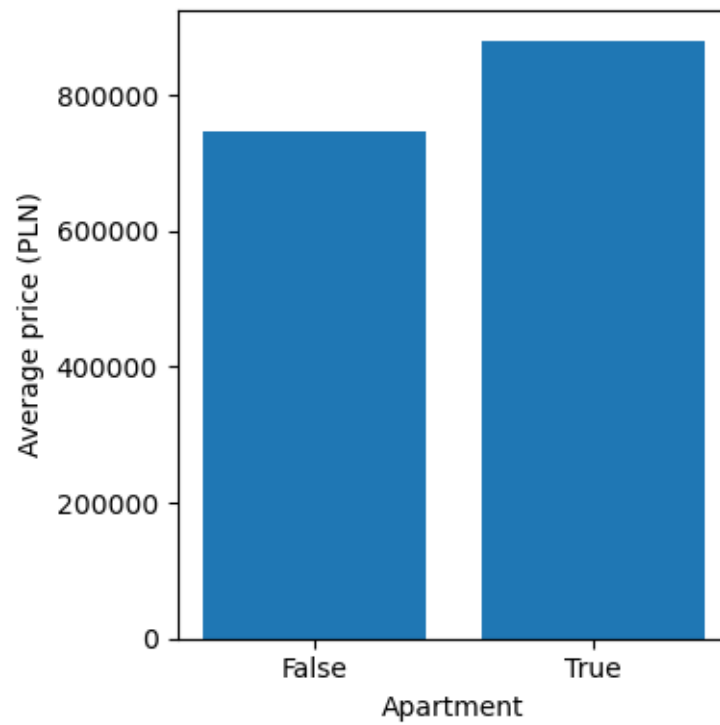
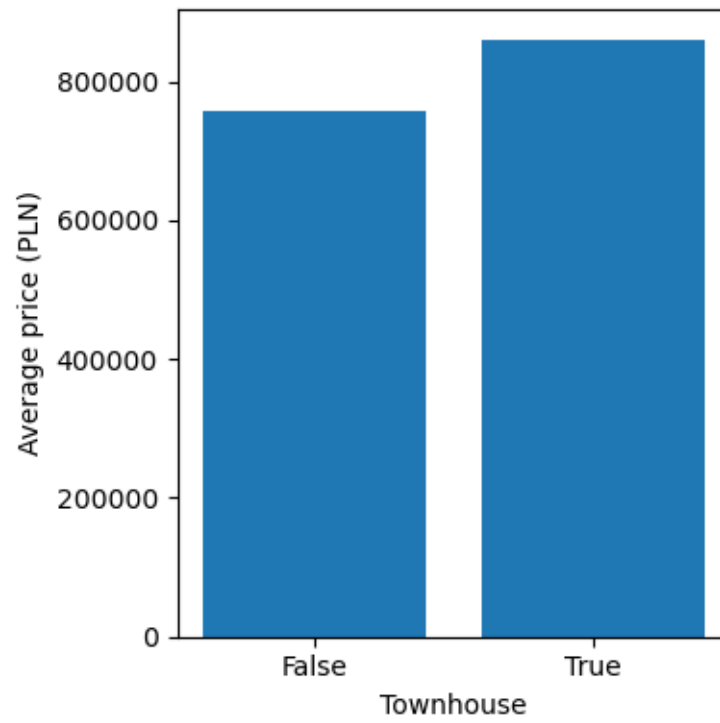
```
[10]: binary = list(data.select_dtypes(bool).columns)
for col in binary:
    group = data[[col, 'Amount']]
    group = group.groupby([col], as_index=False)
    group = group.mean()
    group = group.sort_values('Amount', ascending=False)
    plt.bar(group[col], height=group['Amount'])
    plt.xticks(ticks=group[col].to_list(),
               labels=group[col].to_list())
plt.ylabel('Average price (PLN)')
plt.xlabel(f'{col}')
plt.tight_layout()
```

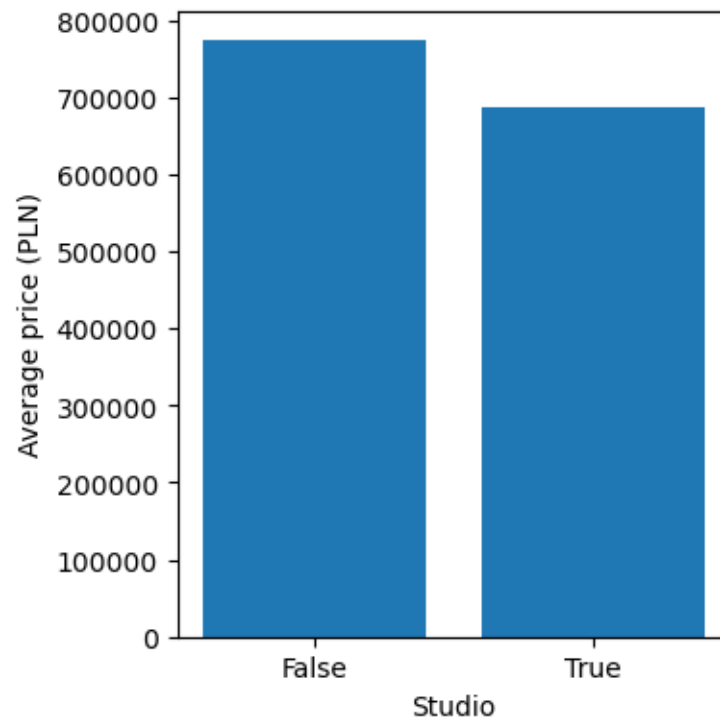
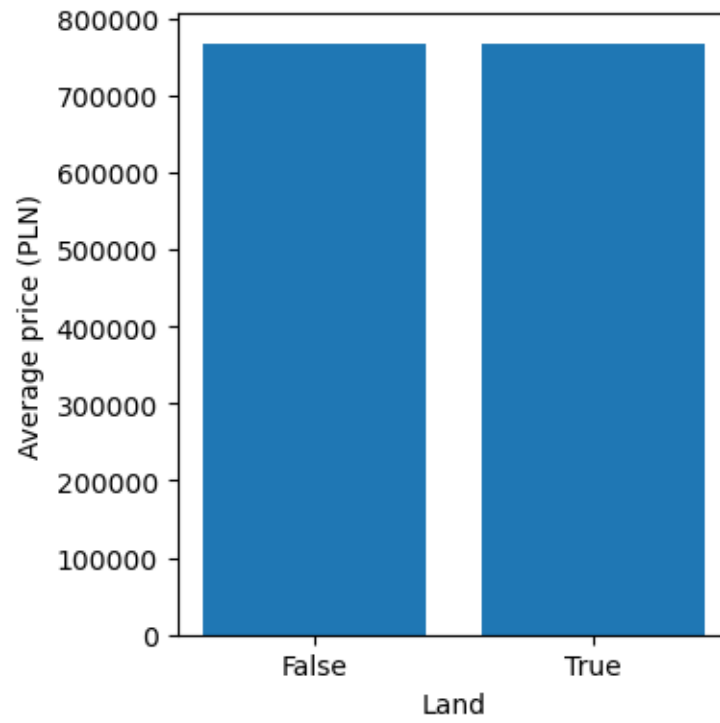
```
plt.show()
```











We check correlation of binary columns with Amount.

```
[11]: C = data[binary].corrwith(data['Amount'])
      C.name = 'Correlation'
      C = C.to_frame()
      C = C.sort_values('Correlation', ascending=False)
      C = C.reset_index()
      C = C.rename(columns={'index': 'Column'})
      C
```

```
[11]:
```

	Column	Correlation
0	Apartment	0.28
1	Terrace	0.18
2	Townhouse	0.18
3	Balcony	0.09
4	Garden	0.07
5	Estate	0.02
6	Land	-0.00
7	Floor	-0.03
8	New	-0.10
9	Studio	-0.13

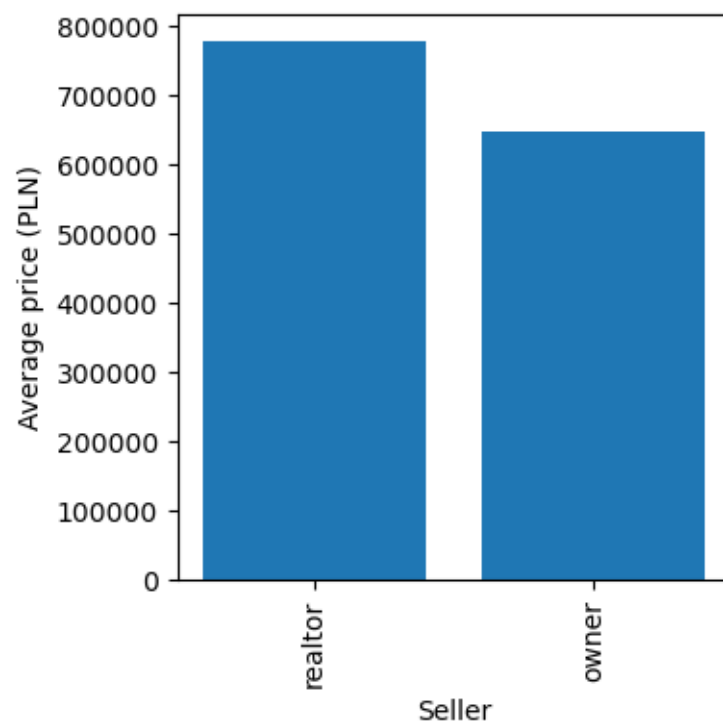
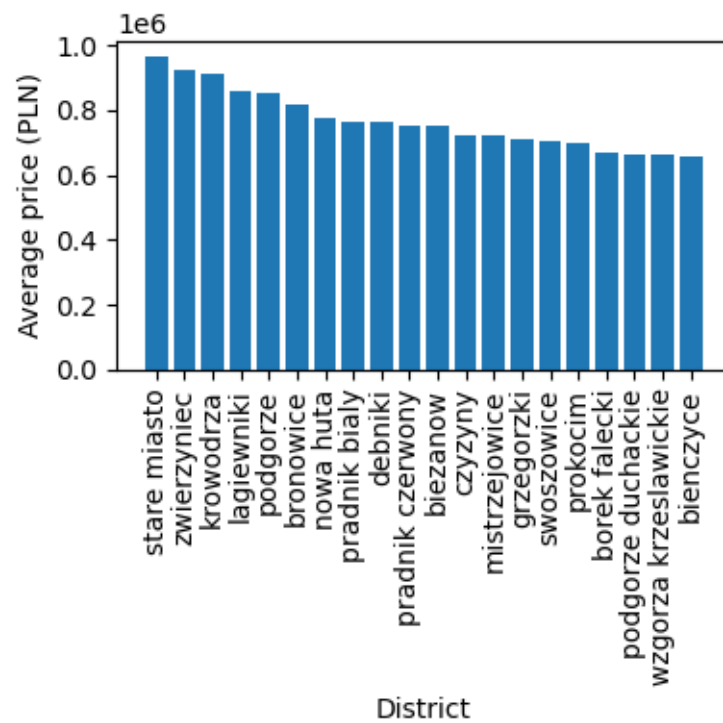
1.6 Categorical features

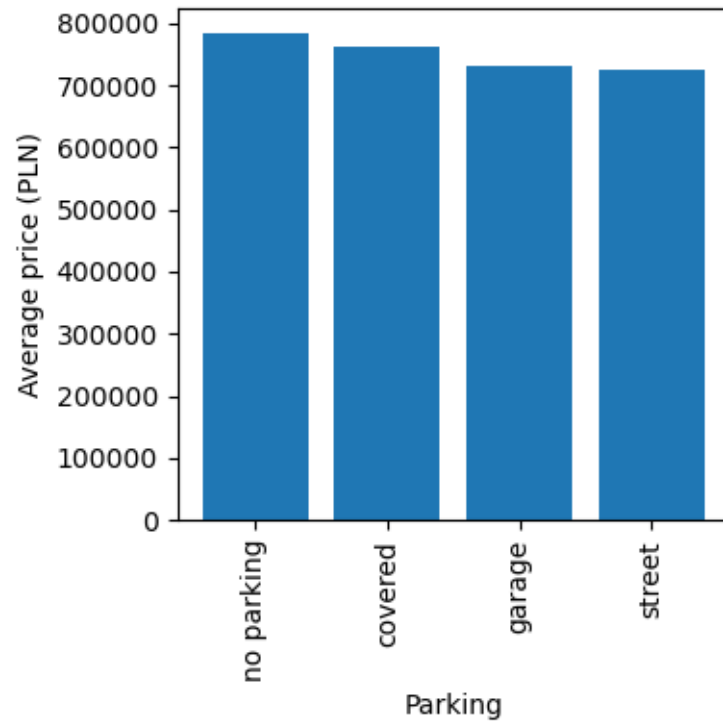
Again, we group the rows and compare averages.

```
[12]: categorical = list(data.select_dtypes('object').columns)
      for col in categorical:
          group = data[[col, 'Amount']]
          group = group.groupby([col], as_index=False)
          group = group.mean()
          group = group.sort_values('Amount', ascending=False)

          plt.bar(group[col], group['Amount'])
          plt.ylabel('Average price (PLN)')
          plt.xlabel(f'{col}')
          plt.xticks(rotation=90)

          plt.tight_layout()
          plt.savefig(f'img/feature_{col.lower()}.png')
          plt.show()
```





[]:

[]:

[]:

[]: