data_wrangling

August 6, 2025

Importing Libraries

[1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

from collections import Counter
from IPython.display import display
from sklearn.impute import KNNImputer
from pylab import rcParams
from pathlib import Path
```

Setup

[2]:
```
# 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
```

Goal

From the property listings Krakow, we would like to create a model to predict flat prices.

Data loading

[3]:
```
data = pd.read_csv('real_estate_dataset.csv')
```

[4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10484 entries, 0 to 10483
Data columns (total 23 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         10484 non-null  object
```

```
 1   City          8815 non-null   object
 2   District      6079 non-null   object
 3   Amount        10484 non-null  float64
 4   Currency      10471 non-null  object
 5   Property      10397 non-null  object
 6   Seller        9661 non-null   object
 7   Area          10355 non-null  float64
 8   Rooms         10484 non-null  int64
 9   Bathrooms     6262 non-null   float64
 10  Parking       3847 non-null   object
 11  Garden        10484 non-null  bool
 12  Balcony       10484 non-null  bool
 13  Terrace       10484 non-null  bool
 14  Floor         10484 non-null  bool
 15  New           10484 non-null  bool
 16  Estate        10484 non-null  bool
 17  Townhouse     10484 non-null  bool
 18  Apartment     10484 non-null  bool
 19  Land          10484 non-null  bool
 20  Studio        10484 non-null  bool
 21  Title         10484 non-null  object
 22  Description   8615 non-null   object
dtypes: bool(10), float64(3), int64(1), object(9)
memory usage: 1.1+ MB
```

First we sort the data in from newest to oldest, forcing rows with missing Date values to be last.

[5]:
```
data = data.sort_values(by='Date',
                        ascending=False,
                        na_position='last',
                        ignore_index=True)
```

Next we assume that the Title column uniquely identifies a listing.

[6]:
```
data = data.drop_duplicates(['Title'], keep='first')
```

After this the shape of the data is:

[7]:
```
print(data.shape)
```

```
(3860, 23)
```

Data Exploration

Check for Missing Values

[8]:
```
missing = data.isnull().sum(axis=0)
missing.name = 'Missing'
missing = missing.to_frame()
missing = missing[missing['Missing'] > 0]
```

```
missing.sort_values('Missing', ascending=False)
```

[8]:

	Missing
Parking	2388
Bathrooms	1511
Description	657
City	605
District	603
Seller	291
Property	84
Area	40
Currency	4

Check Numeric Columns

We see that we have 23 columns at our disposal. We inspect the numeric columns to see what we are dealing with.

[9]:
```
data.describe()
```

[9]:

	Amount	Area	Rooms	Bathrooms
count	3,860.00	3,820.00	3,860.00	2,349.00
mean	822,425.31	82.72	3.03	1.49
std	246,810.86	52.58	1.38	0.72
min	253,705.22	20.00	1.00	1.00
25%	653,873.83	47.58	2.00	1.00
50%	784,263.03	69.59	3.00	1.00
75%	946,935.93	103.15	4.00	2.00
max	3,040,784.63	651.68	6.00	4.00

Check binary columns

We inspect the data to see if binary columns are properly populated and check for imbalances.

[10]:
```
binary = data.select_dtypes(bool).columns.to_list()

for col in binary:
    tmp = data[[col, 'Amount']]
    tmp = tmp.fillna('NaN')
    tmp = tmp.groupby(col, as_index=False)
    tmp = tmp.count()
    tmp = tmp.rename(columns={'Amount': 'Count'})
    tmp = tmp.sort_values('Count', ascending=False)
    tmp = tmp.reset_index(drop=True)
    display(tmp)
```

	Garden	Count
0	False	3094
1	True	766

```
   Balcony  Count
0    False   2513
1     True   1347
```

```
   Terrace  Count
0    False   3410
1     True    450
```

```
   Floor  Count
0  False   2320
1   True   1540
```

```
     New  Count
0  False   2577
1   True   1283
```

```
   Estate  Count
0   False   3268
1    True    592
```

```
   Townhouse  Count
0      False   3515
1       True    345
```

```
   Apartment  Count
0      False   3281
1       True    579
```

```
    Land  Count
0  False   2964
1   True    896
```

```
   Studio  Count
0   False   3556
1    True    304
```

Check categorical columns

We inspect categorical columns to assert that they contain "valid" values. Most of these columns were generated by a script during the scraping and etl phase of the project.

[11]:
```
categorical = data.select_dtypes('object').columns
categorical = categorical.to_list()
omit = ['Title', 'Link', 'Description', 'Date']

for col in categorical:
    if col not in omit:
        tmp = data[['Amount', col]].copy()
        tmp = tmp.fillna('NaN')
        tmp = tmp.groupby(col, as_index=False)
        tmp = tmp.count()
        tmp = tmp.rename(columns={'Amount': 'Count'})
```

```
        tmp = tmp.sort_values('Count', ascending=False)
        tmp = tmp.reset_index(drop=True)
        display(tmp)
```

```
     City  Count
0  kraków   3255
1     NaN    605
```

```
                District  Count
0                    NaN    603
1           stare miasto    176
2               podgorze    174
3                czyzyny    171
4       pradnik czerwony    169
5             swoszowice    169
6               prokocim    168
7               biezanow    167
8             grzegorzki    163
9            zwierzyniec    162
10             krowodrza    161
11             lagiewniki    161
12    podgorze duchackie    161
13             nowa huta    160
14         borek falecki    159
15         pradnik bialy    158
16             bienczyce    157
17         mistrzejowice    156
18             bronowice    155
19               debniki    155
20  wzgorza krzeslawickie    155
```

```
  Currency  Count
0      pln   3856
1      NaN      4
```

```
  Property  Count
0     flat   2779
1    house    997
2      NaN     84
```

```
    Seller  Count
0  realtor   3270
1    owner    299
2      NaN    291
```

```
      Parking  Count
0         NaN   2388
1      garage    606
2      street    569
3  no parking    229
```

```
4      covered      68
```

Data cleaning

We assume that if we know the district, the City is kraków.

[12]:
```
mask = (data['City'].isna() == True) & (data['District'].isna() == False)
data.loc[mask, 'City'] = 'kraków'
```

We extract more Parking information from the property description.

[13]:
```
def extract_parking(x):
    if ('garaż' in x or 'garaz' in x or 'parking' in x) and 'podziemny' in x:
        return 'covered'
    elif ('garaż' in x or 'garaz' in x) and 'podziemny' not in x:
        return 'garage'
    elif 'parking' in x and 'podziemny' not in x:
        return 'street'
    else:
        return 'no parking'
```

[14]:
```
mask = (data['Parking'].isna() == True) & (data['Description'].isna() == False)
data.loc[mask, ['Parking', 'Description']] = data.loc[mask, 'Description'].
 ↪apply(extract_parking)
```

[15]:
```
mask = data['Parking'].isna() == True
data.loc[mask, 'Parking'] = 'no parking'
```

We confirm that we have dealt with all the NaNs in the Parking column.

[16]:
```
print(data['Parking'].isna().sum())
```

```
0
```

Filtering

Next we filter the data according to these rules:

[17]:
```
data = data[data['City'] == 'kraków']
data = data[data['Currency'] == 'pln']
data = data[data['Property'] == 'flat']
data = data[(data['Amount'] >= data['Amount'].quantile(0.025))]
data = data[(data['Amount'] <= data['Amount'].quantile(0.975))]
data = data[(data['Area'] >= data['Area'].quantile(0.01))]
data = data[(data['Area'] <= data['Area'].quantile(0.99))]
data = data[data['District'] != 'unknown']
data = data[data['District'].isna() == False]
data = data[data['Seller'].isna() == False]
data = data[data['Description'].isna() == False]
```

[18]:
```
data = data.reset_index(drop=True)
```

[19]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1767 entries, 0 to 1766
Data columns (total 23 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         1767 non-null   object
 1   City         1767 non-null   object
 2   District     1767 non-null   object
 3   Amount       1767 non-null   float64
 4   Currency     1767 non-null   object
 5   Property     1767 non-null   object
 6   Seller       1767 non-null   object
 7   Area         1767 non-null   float64
 8   Rooms        1767 non-null   int64
 9   Bathrooms    1071 non-null   float64
 10  Parking      1767 non-null   object
 11  Garden       1767 non-null   bool
 12  Balcony      1767 non-null   bool
 13  Terrace      1767 non-null   bool
 14  Floor        1767 non-null   bool
 15  New          1767 non-null   bool
 16  Estate       1767 non-null   bool
 17  Townhouse    1767 non-null   bool
 18  Apartment    1767 non-null   bool
 19  Land         1767 non-null   bool
 20  Studio       1767 non-null   bool
 21  Title        1767 non-null   object
 22  Description  1767 non-null   object
dtypes: bool(10), float64(3), int64(1), object(9)
memory usage: 196.8+ KB
```

Impute missing values

The next step is to fill in missing values for numeric columns Amount Area Rooms and Bathrooms. We use the KNNImputer to accomplish this.

[20]:
```
numeric = list(data.select_dtypes('number').columns)
```

[21]:
```
mask = (data['Bathrooms'].isna() == True | data['Rooms'].isna())
missing = data[numeric]

imputer = KNNImputer(n_neighbors=5)
imputer.fit(missing)
```

```
missing = imputer.transform(missing)
missing = pd.DataFrame(missing, columns=numeric)

for col in numeric:
    data[col] = missing[col]

for col in numeric:
    data[col] = data[col].apply(lambda x: round(x))
```

[22]:
```
print(data.shape)
```

```
(1767, 23)
```

Save data

Verify that there are no NaNs in data.

[23]:
```
data.isnull().sum().sum()
```

[23]:
```
np.int64(0)
```

Remove columns that will not be used further.

[24]:
```
data = data.drop(['Title',
                  'Description',
                  'Property',
                  'City',
                  'Currency',
                  'Date'], axis=1)
```

Take a last peek at the data.

[25]:
```
data.head()
```

[25]:

	District	Amount	Seller	Area	Rooms	Bathrooms	Parking	Garden	Balcony	Terrace	Floor	New	Estate	Townhouse	Apartment	Land	Studio
0	biezanow	536505	realtor	22	1	1	no parking	False	False	False	False	False	False	False	False	False	False
1	bienczyce	646975	realtor	46	2	1	no parking	True	True	False	False	False	False	True	False	False	False
2	lagiewniki	816233	realtor	37	1	1	no parking	False	False	True	False	True	False	False	False	False	False
3	zwierzyniec	1009826	realtor	55	2	2	no parking	True	False	False	False	True	False	True	False	False	False
4	bronowice	733546	realtor	67	2	1	garage	True	False	False	False	True	False	False	False	False	False

[26]:
```
data.describe()
```

[26]:

	Amount	Area	Rooms	Bathrooms
count	1,767.00	1,767.00	1,767.00	1,767.00
mean	767,417.36	67.45	2.71	1.33
std	173,627.88	30.57	1.26	0.56
min	453,475.00	22.00	1.00	1.00
25%	636,175.50	44.00	2.00	1.00
50%	748,405.00	62.00	3.00	1.00
75%	879,074.00	86.00	3.00	2.00
max	1,243,885.00	171.00	6.00	4.00

Save it for further analysis.

[27]:
```
data.to_csv('cleaned_real_estate.csv', index=False)
```