Submission_Arshdeep

February 15, 2025

0.1 Data Ingestion

0.1.1 Imports and Data Download

[1]:
```
# !python -m spacy download en_core_web_sm
```

[2]:
```
import requests
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from collections import Counter
import re
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from wordcloud import WordCloud
# import spacy
# from tqdm import tqdm
```

[3]:
```
# FOLDER_ID = '13AZpzOC2HgYxUnChGCk8Q9g3mIJ8Q-Ku'
# API_KEY = "AIzaSyA8RPhv9HO9C3if71arj7F6uFtSs_IobYE"

# # List files in the folder
# url = f"https://www.googleapis.com/drive/v3/files?
↪q='{FOLDER_ID}'+in+parents&key={API_KEY}"
# response = requests.get(url)
# files = response.json()
# print(files)  # Print file details
```

[4]:
```
# # Download train_set.csv
# file_id = "1UeCuTCODvysuVNzQOhr7n9_0Q1pj1Baw"  # Get this from the files list
```

```
# download_url = f"https://www.googleapis.com/drive/v3/files/{file_id}?
↪alt=media&key={API_KEY}"
# file_response = requests.get(download_url)

# # Save the file
# with open("train_set", "wb") as f:
#     f.write(file_response.content)
```

[5]:
```
# # Download test_set.csv
# file_id = "12IdfMBzYpRc8Uyy5vsp5G4CZx0ONrX8r"  # Get this from the files list
# download_url = f"https://www.googleapis.com/drive/v3/files/{file_id}?
↪alt=media&key={API_KEY}"
# file_response = requests.get(download_url)

# # Save the file
# with open("test_set", "wb") as f:
#     f.write(file_response.content)
```

0.2 Exploratory Data Analysis (EDA)

[6]:
```
# Load the CSV file
train_df = pd.read_csv("train_set")

# Display the first 5 rows
train_df.head()
```

[6]:
```
           id               region  price       type  sqfeet  beds  baths  \
0  7049965813        orange county   2632  apartment    1080     2    2.0
1  7036046796       visalia-tulare   1160  apartment     768     2    1.0
2  7037856890             portland   1262  apartment    1075     2    1.0
3  7046933042              boulder   1861  apartment    1076     2    2.0
4  7048650961  sioux falls / SE SD    626  apartment     720     1    1.0

   cats_allowed  dogs_allowed  smoking_allowed  wheelchair_access  \
0             1             1                0                  0
1             1             1                1                  0
2             1             1                0                  0
3             1             1                0                  0
4             1             1                0                  0

   electric_vehicle_charge  comes_furnished  laundry_options  \
0                        0                0      w/d in unit
1                        0                0  laundry on site
2                        0                0  laundry on site
3                        1                0      w/d in unit
4                        0                0      w/d in unit
```

```
     parking_options                                        description  \
0    attached garage  To schedule a tour We now book our tour appoin…
1            carport  Oak View is just minutes from Highway 99 and b…
2 off-street parking  ***Please Pardon Our Dust*** Do you want three…
3    detached garage  Luna Bella  Pet Friendly  Community in Lafayet…
4    detached garage  Creekstone Falls\t                           Prop…

      lat      long state
0 33.8123 -117.8530    ca
1 36.3008 -119.3440    ca
2 45.5273 -122.4800    or
3 39.9744 -105.0850    co
4 43.5179  -96.7924    sd
```

[7]:
```python
test_df = pd.read_csv("test_set")
test_df.head()
```

[7]:
```
          id                 region price       type sqfeet  beds  baths  \
0 7027550434   worcester / central MA  1750  apartment    936     2    1.0
1 7033922487      mcallen / edinburg   850  apartment   1200     2    2.5
2 7045296557  fort collins / north CO  1500  apartment   1029     2    1.0
3 7031700539             indianapolis   899  apartment    856     2    1.0
4 7048945590               cincinnati   595  apartment    350     0    1.0

   cats_allowed  dogs_allowed  smoking_allowed  wheelchair_access  \
0            1             0                1                  0
1            0             0                0                  0
2            1             1                0                  0
3            1             1                1                  0
4            0             0                0                  0

   electric_vehicle_charge  comes_furnished     laundry_options  \
0                        0                0                 NaN
1                        0                0         w/d in unit
2                        0                0         w/d in unit
3                        0                0      laundry in bldg
4                        0                0  no laundry on site

     parking_options                                        description  \
0                NaN  VIEW OUR WEBSITE:  https://www.winncompanies.c…
1 off-street parking  2 bedrooms, 2.5 baths, refrigerator, stove, wa…
2 off-street parking  Brand New 2 bed 1 bath apartment will go FAST …
3                NaN  Check out what these 2 bedrooms @ Teal Run hav…
4 off-street parking  Completely renovated with an urban flare. 3rd …

      lat      long state
0 42.2526  -71.8499    ma
```

```
1  26.2154  -98.2359    tx
2  40.3849 -105.0920    co
3  39.7886  -85.9779    in
4  39.1275  -84.5350    oh
```

[8]:
```
train_df.shape
```

[8]:
```
(346479, 19)
```

[9]:
```
test_df.shape
```

[9]:
```
(38498, 19)
```

[10]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 346479 entries, 0 to 346478
Data columns (total 19 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   id                       346479 non-null  int64
 1   region                   346479 non-null  object
 2   price                    346479 non-null  int64
 3   type                     346479 non-null  object
 4   sqfeet                   346479 non-null  int64
 5   beds                     346479 non-null  int64
 6   baths                    346479 non-null  float64
 7   cats_allowed             346479 non-null  int64
 8   dogs_allowed             346479 non-null  int64
 9   smoking_allowed          346479 non-null  int64
 10  wheelchair_access        346479 non-null  int64
 11  electric_vehicle_charge  346479 non-null  int64
 12  comes_furnished          346479 non-null  int64
 13  laundry_options          275399 non-null  object
 14  parking_options          219914 non-null  object
 15  description              346477 non-null  object
 16  lat                      344728 non-null  float64
 17  long                     344728 non-null  float64
 18  state                    346479 non-null  object
dtypes: float64(3), int64(10), object(6)
memory usage: 50.2+ MB
```

[11]:
```
train_df.describe()
```

[11]:
```
                 id         price        sqfeet           beds          baths  \
count  3.464790e+05  3.464790e+05  3.464790e+05  346479.000000  346479.000000
mean   7.040981e+09  1.615492e+03  1.067028e+03       1.904468       1.481110
std    8.798642e+06  6.586159e+04  2.018564e+04       3.256987       0.618514
```

```
min    7.003808e+09  0.000000e+00  0.000000e+00        0.000000        0.000000
25%    7.035975e+09  8.060000e+02  7.500000e+02        1.000000        1.000000
50%    7.043314e+09  1.039000e+03  9.500000e+02        2.000000        1.000000
75%    7.048429e+09  1.395000e+03  1.150000e+03        2.000000        2.000000
max    7.051292e+09  2.170191e+07  8.388607e+06     1100.000000       75.000000

        cats_allowed   dogs_allowed  smoking_allowed  wheelchair_access  \
count  346479.000000  346479.000000    346479.000000      346479.000000
mean        0.726780       0.707847         0.731987           0.081939
std         0.445614       0.454753         0.442925           0.274272
min         0.000000       0.000000         0.000000           0.000000
25%         0.000000       0.000000         0.000000           0.000000
50%         1.000000       1.000000         1.000000           0.000000
75%         1.000000       1.000000         1.000000           0.000000
max         1.000000       1.000000         1.000000           1.000000

       electric_vehicle_charge  comes_furnished            lat           long
count            346479.000000    346479.000000  344728.000000  344728.000000
mean                  0.012806         0.048040      37.237436     -92.706870
std                   0.112437         0.213852       5.542563      16.522223
min                   0.000000         0.000000     -43.533300    -163.894000
25%                   0.000000         0.000000      33.465200    -100.784000
50%                   0.000000         0.000000      37.658000     -87.772500
75%                   0.000000         0.000000      41.141000     -81.179600
max                   1.000000         1.000000     102.036000     172.633000
```

0.2.1 1.1 Checking variance of numerical columns

[12]:
```python
numerical_cols = ["price", "sqfeet", "beds", "baths", "lat", "long"]
for col in numerical_cols:
    print("Variance for",col,"column :", train_df[col].var())
```

```
Variance for price column : 4337748577.786544
Variance for sqfeet column : 407460235.51401496
Variance for beds column : 10.607963203245545
Variance for baths column : 0.38255925261972035
Variance for lat column : 30.720008881592843
Variance for long column : 272.98384804658565
```

Key Takeaways - Very high variance → Expect outliers and wide range of values. - Most properties have similar bathroom & bedroom counts. - Geographic spread is uneven, possibly indicating some regions dominate the dataset.

0.2.2 1.2 Checking skewness of numerical columns

[13]:
```
for col in numerical_cols:
    print("Skewness for", col, "column:", train_df[col].skew())
```

```
Skewness for price column: 235.39303545130664
Skewness for sqfeet column: 388.96706906430643
Skewness for beds column: 304.2929719312646
Skewness for baths column: 10.648135085840252
Skewness for lat column: 0.249342430946829
Skewness for long column: -0.05552793176059056
```

Key Takeaways - Price, Sqfeet, Beds, and Baths are all highly right-skewed. - Latitude and Longitude are nearly symmetrical.

0.2.3 1.3 Checking for na values

[14]:
```
# missing values
train_df.isna().sum()
```

[14]:
```
id                             0
region                         0
price                          0
type                           0
sqfeet                         0
beds                           0
baths                          0
cats_allowed                   0
dogs_allowed                   0
smoking_allowed                0
wheelchair_access              0
electric_vehicle_charge        0
comes_furnished                0
laundry_options            71080
parking_options           126565
description                    2
lat                         1751
long                        1751
state                          0
dtype: int64
```

Key Takeaways: - laundry_options has 71,080 missing values, its Categorical, we can fill with mode (most common value) - parking_options has 126,565 missing values, its Categorical, we can fill with mode (most common parking value: off-street parking) - description has 2 missing values, we can drop them - lat has 1,751 missing values, its Numeric, we can fill it with region-wise median - long has 1,751 missing values, its Numeric, we can fill it with region-wise median

0.2.4 1.4 Checking for duplicate rows

[15]:
```
train_df.duplicated().sum()
```

[15]: 0

No duplicate rows

0.2.5 1.5 Checking value counts

[16]:
```
for col in ["region", "price", "type", "sqfeet", "beds", "baths",␣
↪"laundry_options", "parking_options", "state"]:
    print(train_df[col].value_counts(),"\n")
```

```
region
jacksonville      3830
columbus          3366
rochester         3306
fayetteville      3300
jackson           3299
                  …
southwest MS        11
southwest TX         9
st louis             7
fort smith, AR       5
kansas city          3
Name: count, Length: 404, dtype: int64

price
750         3782
850         3602
800         3602
1200        3470
950         3267
            …
13995          1
26             1
3943           1
11621360       1
3516           1
Name: count, Length: 3877, dtype: int64

type
apartment          286195
house               29978
townhouse           14329
condo                5598
duplex               4496
manufactured         3821
```

```
cottage/cabin         794
loft                  636
flat                  473
in-law                151
land                    6
assisted living         2
Name: count, dtype: int64

sqfeet
1000    9057
900     7633
1100    6333
800     6223
700     5982
        …
332        1
3260       1
3410       1
2822       1
2848       1
Name: count, Length: 3204, dtype: int64

beds
2       157981
1       105402
3        60415
4        10439
0         9857
5         2099
6          211
7           44
8           28
1100         2
1000         1
Name: count, dtype: int64

baths
1.0     178202
2.0     121309
1.5      24600
2.5      11858
3.0       5030
0.0       2821
4.0       1326
3.5        918
4.5        210
5.0        118
5.5         50
```

```
6.0         23
6.5          4
7.0          3
75.0         2
7.5          2
8.0          1
25.0         1
8.5          1
Name: count, dtype: int64

laundry_options
w/d in unit           118694
w/d hookups            67873
laundry on site        53006
laundry in bldg        32545
no laundry on site      3281
Name: count, dtype: int64

parking_options
off-street parking    115621
attached garage        36578
carport                35132
detached garage        15233
street parking         14331
no parking              2870
valet parking            149
Name: count, dtype: int64

state
ca    29821
fl    28696
tx    28052
nc    16732
mi    13091
ga    12426
oh    11585
tn    10392
co    10227
va     9953
ny     9057
sc     8943
pa     8932
il     8727
al     7332
or     7238
ks     7103
ia     6784
md     6712
```

```
mn     6690
la     6549
wa     6364
az     6092
wi     6005
in     5806
ok     5192
nj     5120
ky     4859
ut     4699
ms     4484
ma     4425
id     4010
ct     3387
nd     3063
ar     2858
nm     2609
nv     2550
ne     2447
dc     2245
ak     1949
mo     1941
de     1830
ri     1709
hi     1654
sd     1602
nh     1582
mt     1197
wv      740
vt      467
me      381
wy      170
Name: count, dtype: int64
```

Key Takeaways

Region - Some regions have very few listings, These may not be reliable for model predictions due to limited data. - Some region names have inconsistent formatting ("fort smith, AR"), this included state, which is a different feature, need standardization

price - Some extreme values exist (11621360, 26) → These are likely outliers and should be examined.

type (Property type) - Apartments dominate (82% of listings) → Other property types contribute much less. - Rare categories (land, assisted living) have very few listings → These might be removed or grouped into an "Other" category. - Potential Issue: One-hot encoding could create sparse data due to many low-frequency categories.

sqfeet - Could contain extreme outlier values, need to checked with boxplot and log graph, and removed/corrected

beds and **baths** - Extreme values (beds=1000, baths=75) are unrealistic and should be removed as outliers - Some of the properties with zero beds, are outliers

[17]:
```
train_df[(train_df["beds"] == 1000)]
```

[17]:
```
               id      region  price       type  sqfeet  beds  baths  \
25290  7036731908  youngstown    550  apartment     250  1000   25.0

       cats_allowed  dogs_allowed  smoking_allowed  wheelchair_access  \
25290             0             0                0                  0

       electric_vehicle_charge  comes_furnished laundry_options  \
25290                        0                0             NaN

      parking_options description      lat      long state
25290  street parking   2 bedroom  41.0252 -80.6687    oh
```

[18]:
```
train_df[(train_df["beds"] == 1100)]
```

[18]:
```
                id   region  price       type  sqfeet  beds  baths  \
51403   7046224868  chicago   2449  apartment    1000  1100   75.0
153122  7045590325  chicago   2449  apartment    1000  1100   75.0

        cats_allowed  dogs_allowed  smoking_allowed  wheelchair_access  \
51403              0             0                1                  0
153122             0             0                1                  0

        electric_vehicle_charge  comes_furnished laundry_options  \
51403                         0                0     w/d in unit
153122                        0                0     w/d in unit

       parking_options                                        description  \
51403          carport  Furnished or Unfurnished Units Includes Parkin…
153122         carport  Furnished or Unfurnished Units Includes Parkin…

            lat    long state
51403   42.0195 -87.665    il
153122  42.0195 -87.665    il
```

These above are outliers, they needs to be removed

[19]:
```
train_df[(train_df['beds'] == 0)].head(2)
```

[19]:
```
            id           region  price       type  sqfeet  beds  baths  \
14  7036089161  college station    589  apartment     480     0    1.0
45  7048022092          lincoln    499  apartment     495     0    1.0
```

```
    cats_allowed  dogs_allowed  smoking_allowed  wheelchair_access  \
14             1             1                1                  0
45             1             0                0                  0

    electric_vehicle_charge  comes_furnished  laundry_options  \
14                        0                0              NaN
45                        0                0  laundry in bldg

       parking_options                                        description  \
14                 NaN  Join Us Today! @ The Gables  Located less than…
45  off-street parking  www.LNKhousing.com |  show contact info  |  sh…

        lat     long state
14  30.6190 -96.3185    tx
45  40.7893 -96.6938    ne
```

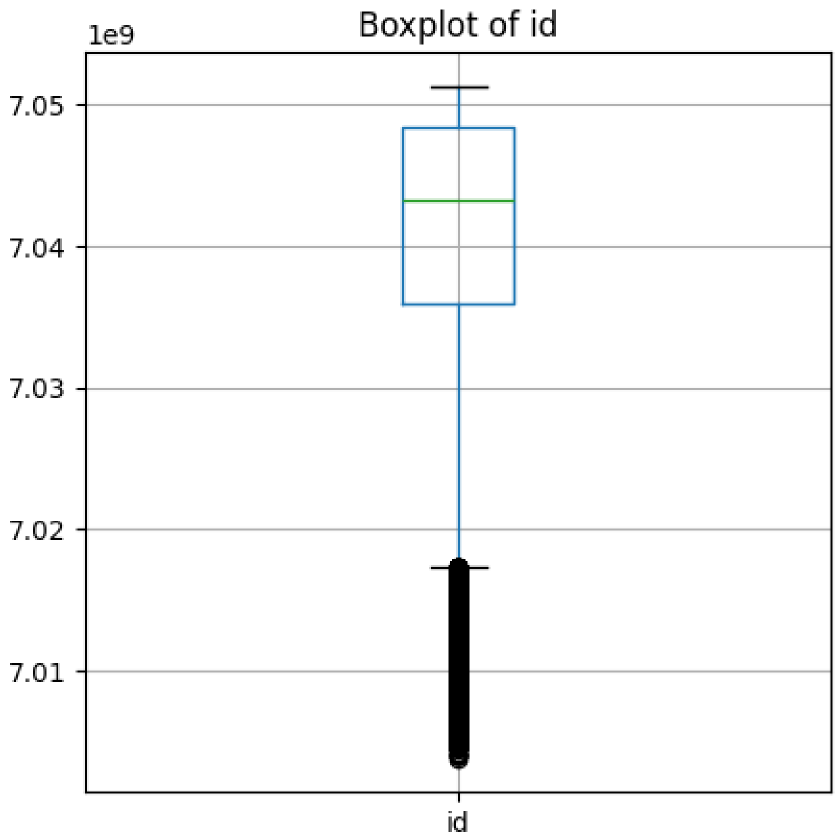
Zero beds properties are outliers, needs to removed

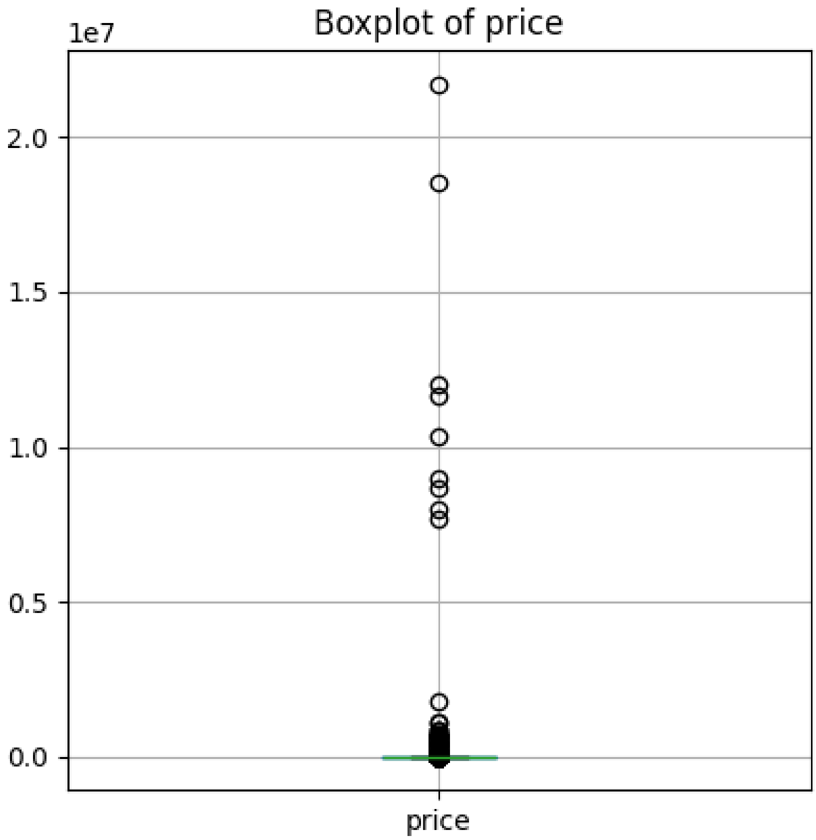
0.2.6 1.6 Detecting outliers using Boxplots (IQR method)

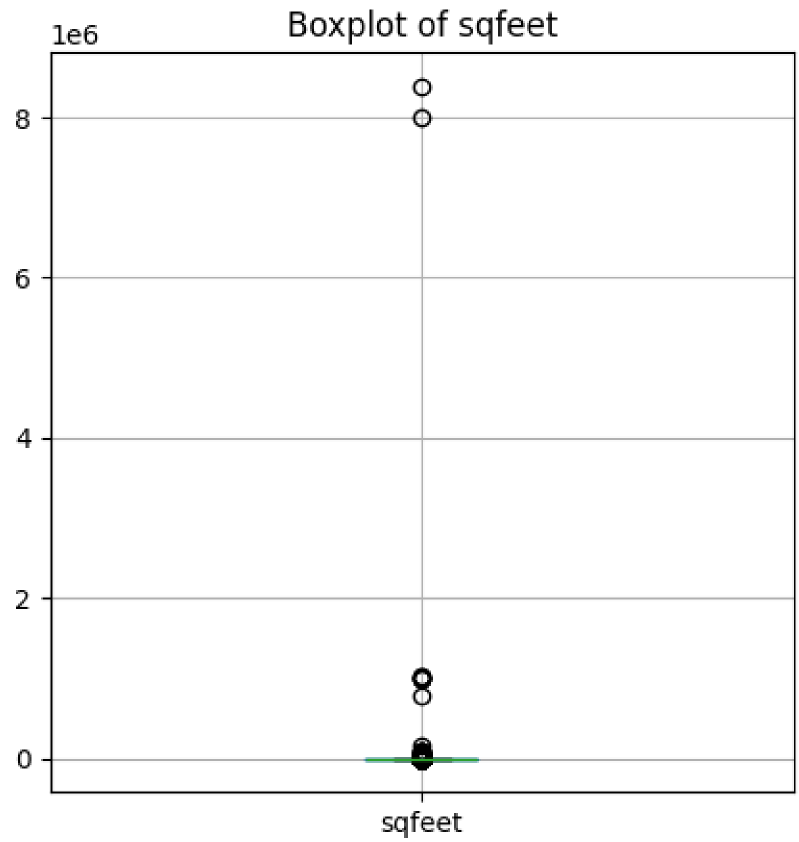
[20]:
```python
# Function to "Plot Box Plot" and it works only for the datatype of integer and␣
 ↪float columns .
def PBP (data):
    for col in data.select_dtypes(include=['int', 'float']).columns:
        fig, ax = plt.subplots(figsize=(5, 5))
        ax.set_title(f'Boxplot of {col}')
        bp=data.boxplot(column=col)
        plt.show()
        plt.close(fig)
```

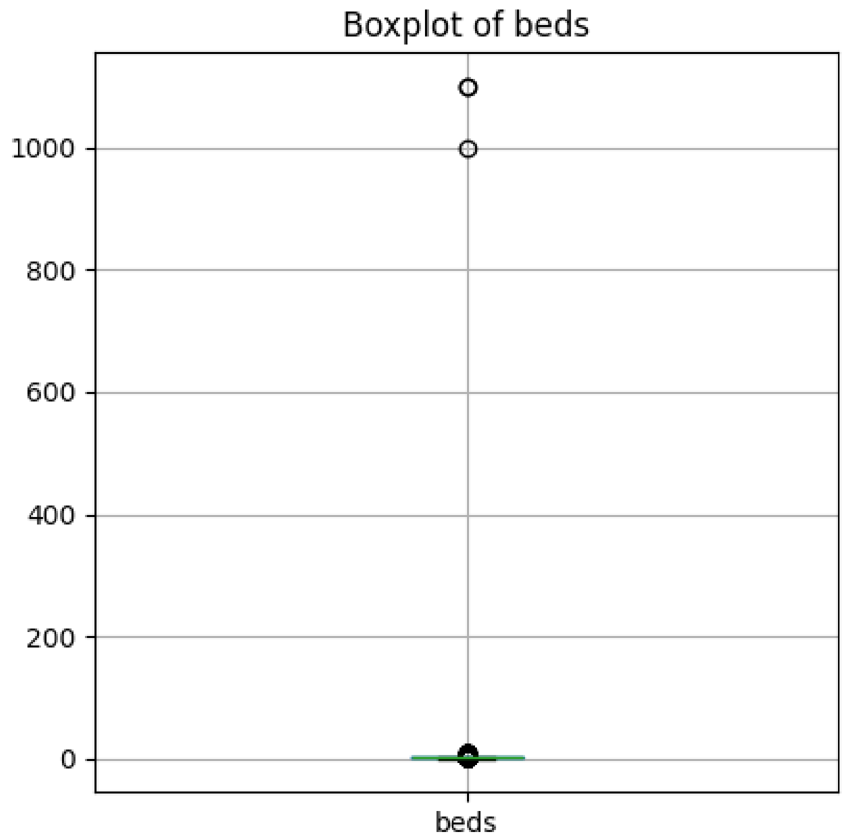
[21]:
```python
# sorted_price = train_df["price"].sort_values(ascending=False)
# sorted_price
```

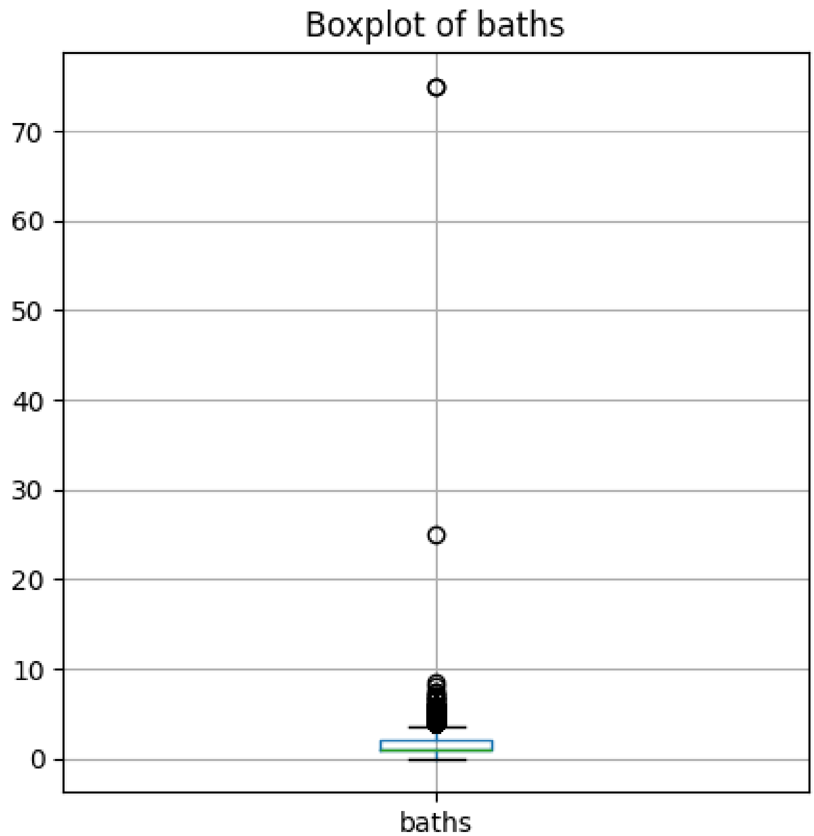
[22]:
```python
PBP(train_df)
```

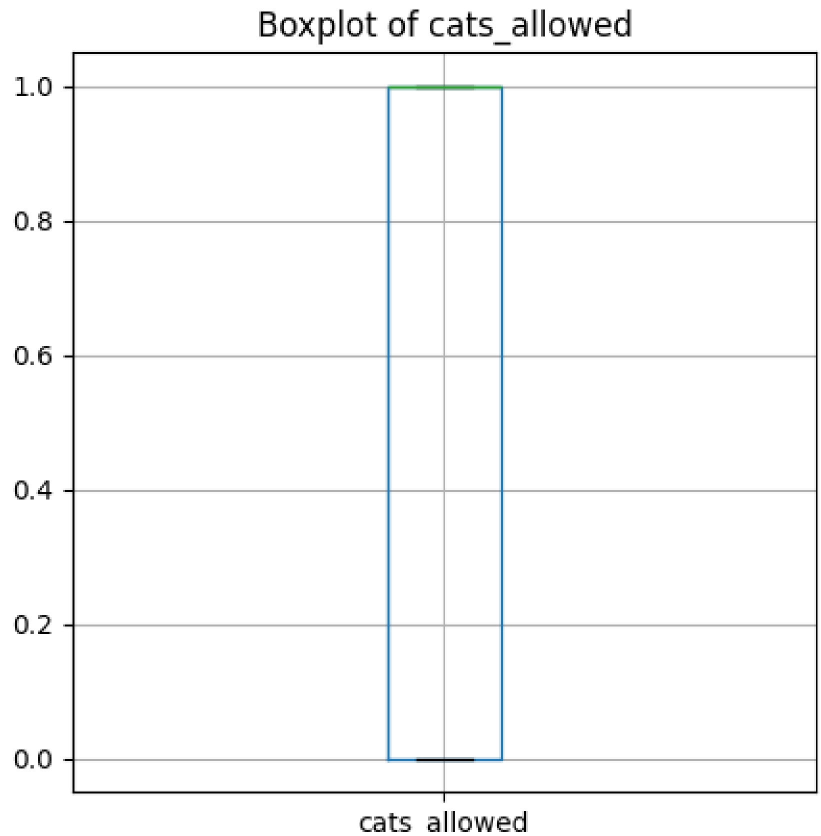

Boxplot of id
1e9
7.05
7.04
7.03
7.02
7.01
id

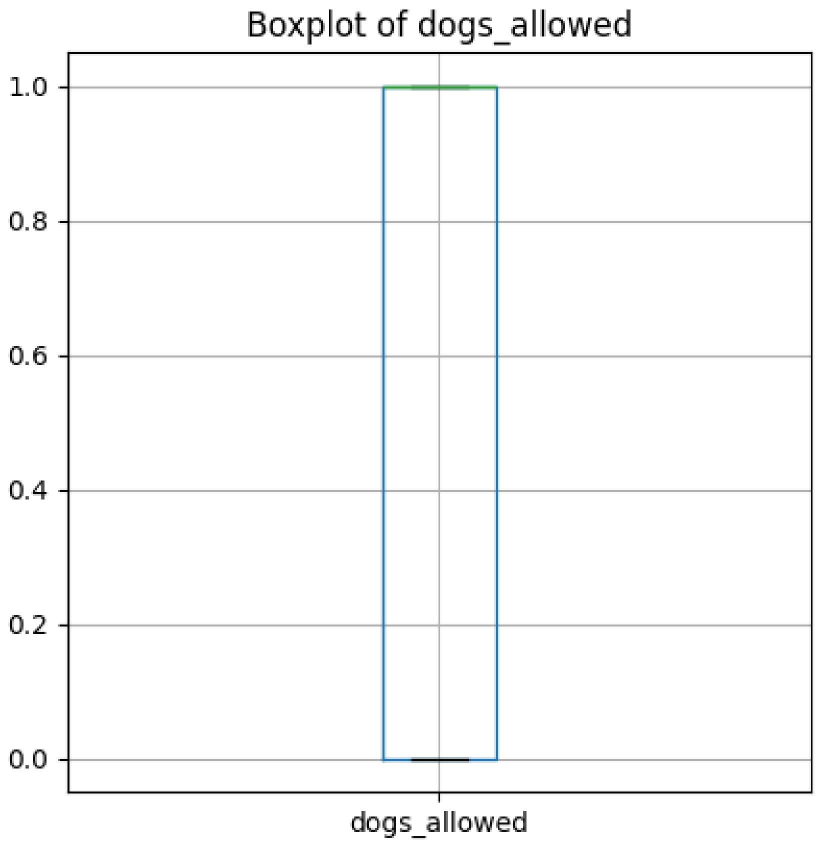

Boxplot of price
1e7
2.0
1.5
1.0
0.5
0.0
price

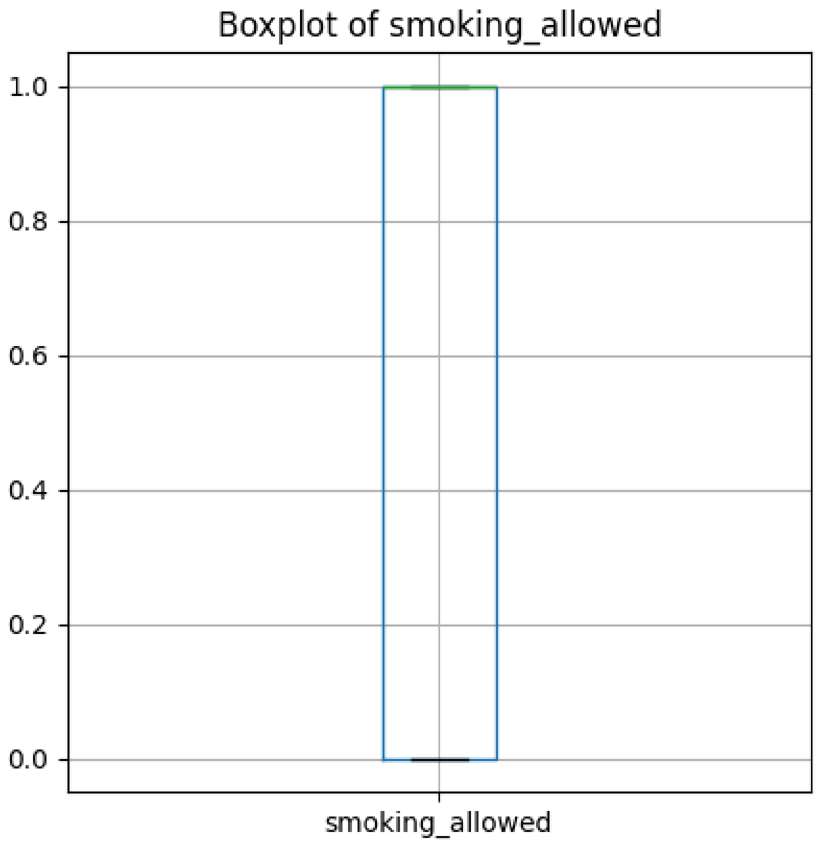

Boxplot of sqfeet
1e6
8
6
4
2
0
sqfeet

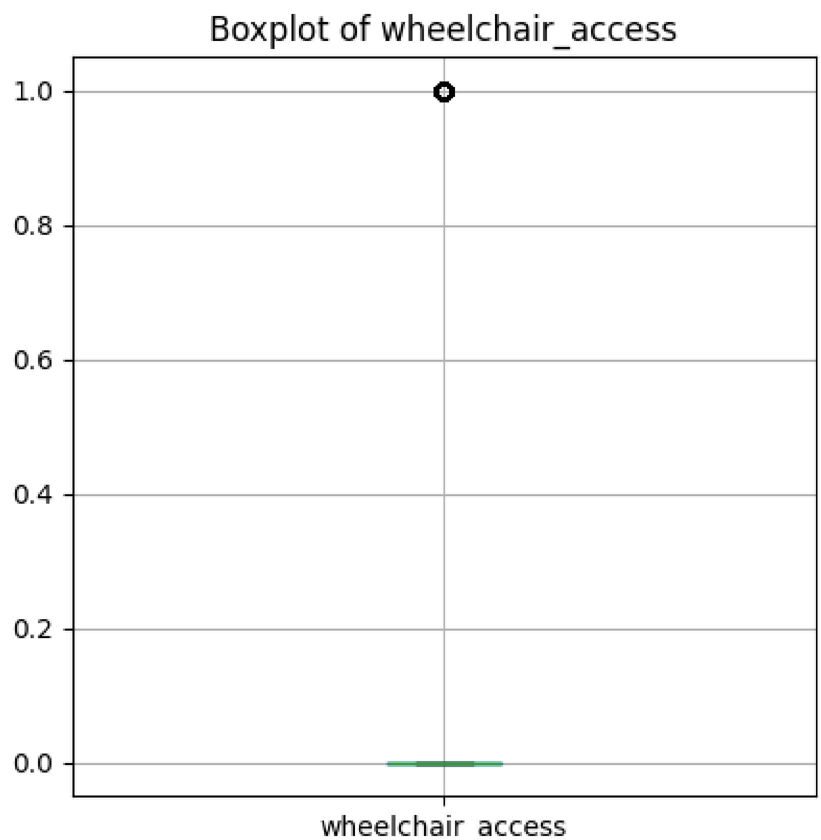

Boxplot of beds
1000
800
600
400
200
0
beds

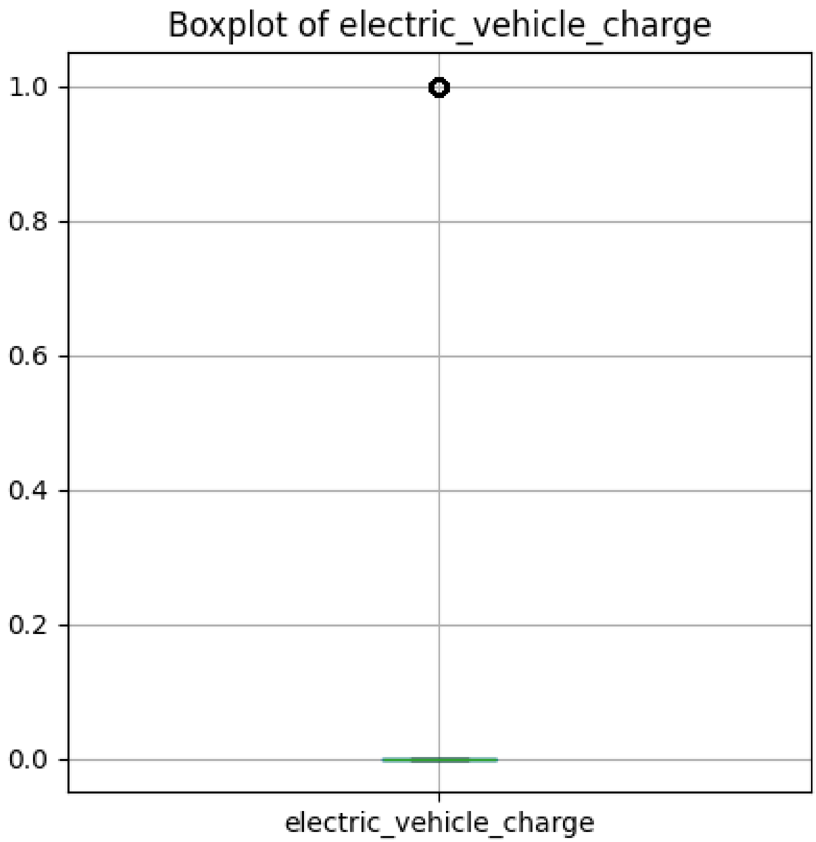

Boxplot of baths
70
60
50
40
30
20
10
0
baths

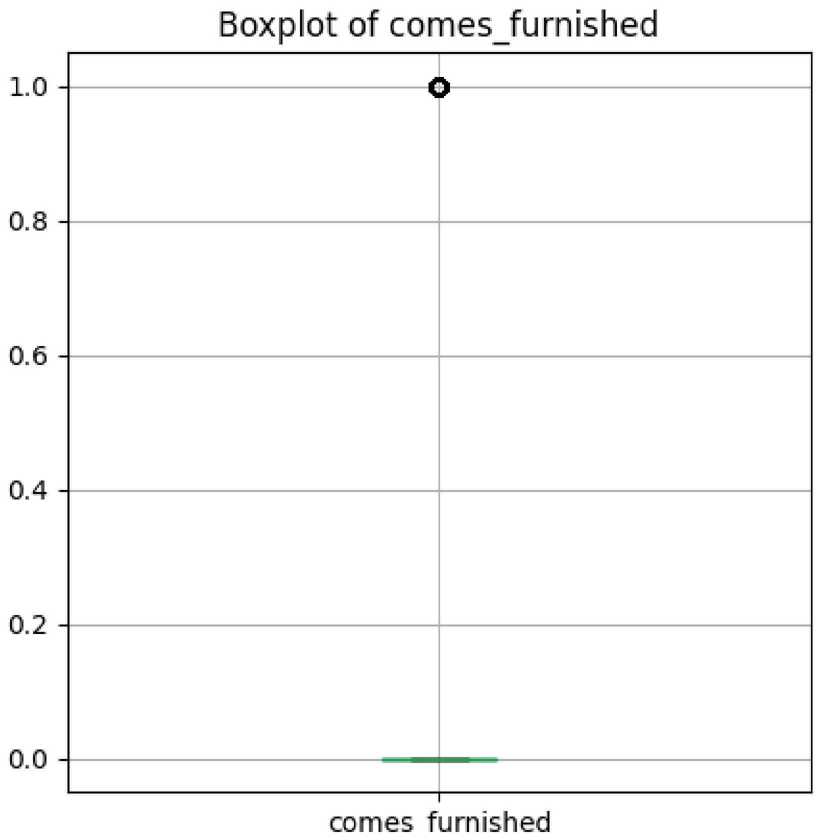

Boxplot of cats_allowed
1.0
0.8
0.6
0.4
0.2
0.0
cats_allowed

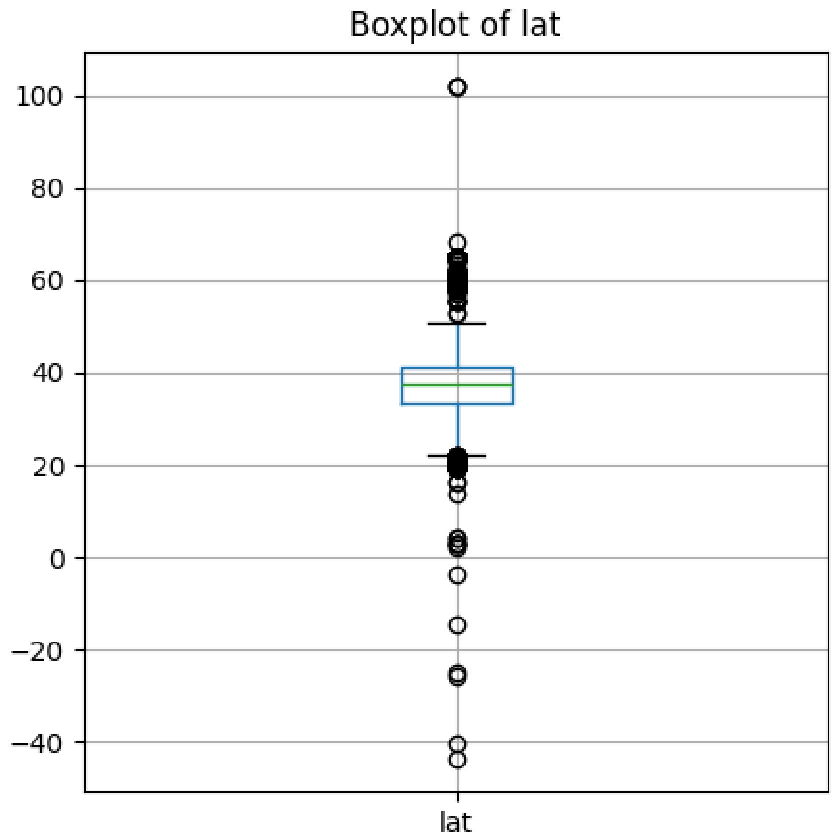

Boxplot of dogs_allowed
1.0
0.8
0.6
0.4
0.2
0.0
dogs_allowed

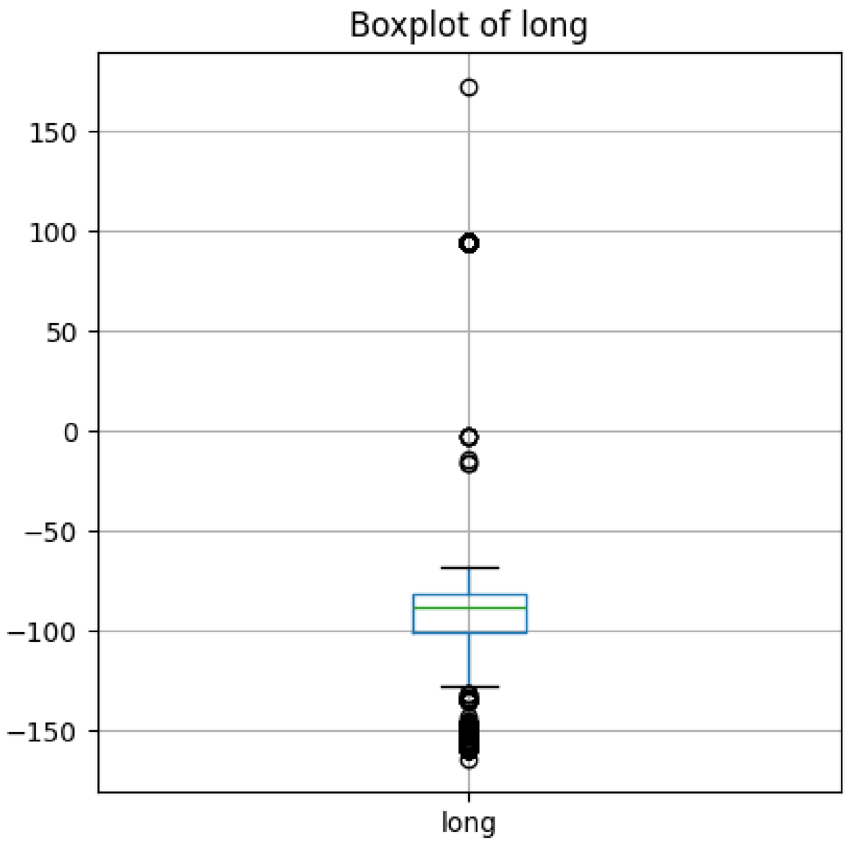

Boxplot of smoking_allowed
1.0
0.8
0.6
0.4
0.2
0.0
smoking_allowed


Boxplot of wheelchair_access
1.0
0.8
0.6
0.4
0.2
0.0
wheelchair_access


Boxplot of electric_vehicle_charge
1.0
0.8
0.6
0.4
0.2
0.0
electric_vehicle_charge


Boxplot of comes_furnished
1.0
0.8
0.6
0.4
0.2
0.0
comes_furnished


Boxplot of lat
100
80
60
40
20
0
−20
−40
lat



Key Takeaways for outliers - Price is over 500k per month for some properties and 0 - SqFeet for 2 properties is extremely large (skewed), and some of them are 0 - Properties with 0 beds, and 1000+ beds are outliers - Properties with 25+ baths are outliers, they are 1000sqfeet, so that's not physically possible

0.2.7 1.7 Histogram/Distribution plots

[23]:
```
# Function to Plot Distribution Plot
def PDP(data):
    columns = [col for col in data.columns]
    colors = sns.color_palette("dark", len(columns))

    for i, col in enumerate(data.select_dtypes(include=['int', 'float']).
 ↪columns):
        if data[col].value_counts().count() <= 10000:
            sns.histplot(data[col], kde=True, bins=15, color=colors[i %␣
 ↪len(colors)])
            plt.title(f'Distribution of {col}')
            plt.show()
```

[24]:
```
# PDP(train_df)
```

0.2.8 1.8 Analyzing description column feature

[27]:
```
def preprocess_text(text):
    if isinstance(text, str):  # Check if the value is a string
        text = text.lower()  # Convert to lowercase
        text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
        return text
    else:
        return ""  # Return empty string for NaN or non-string values

# description_cleaned = train_df['description'].dropna().apply(preprocess_text)
# description_cleaned
```

[28]:
```
# # Extract Key Terms using TF-IDF (removes standard English stopwords)
# vectorizer = TfidfVectorizer(stop_words="english", max_features=100)
# tfidf_matrix = vectorizer.fit_transform(description_cleaned)

# # Get feature names (important words)
# top_words = vectorizer.get_feature_names_out()

# # Combine text and remove additional stopwords
# text = " ".join(str(desc) for desc in description_cleaned)
# wordcloud = WordCloud(
#     width=800, height=400, background_color="white", max_words=200,
#     stopwords=vectorizer.get_stop_words()
# ).generate(text)

# # Display Word Cloud
# plt.figure(figsize=(10, 5))
# plt.imshow(wordcloud, interpolation="bilinear")
# plt.axis("off")
# plt.title("Cleaned Word Cloud of Description Text")
# plt.show()
```

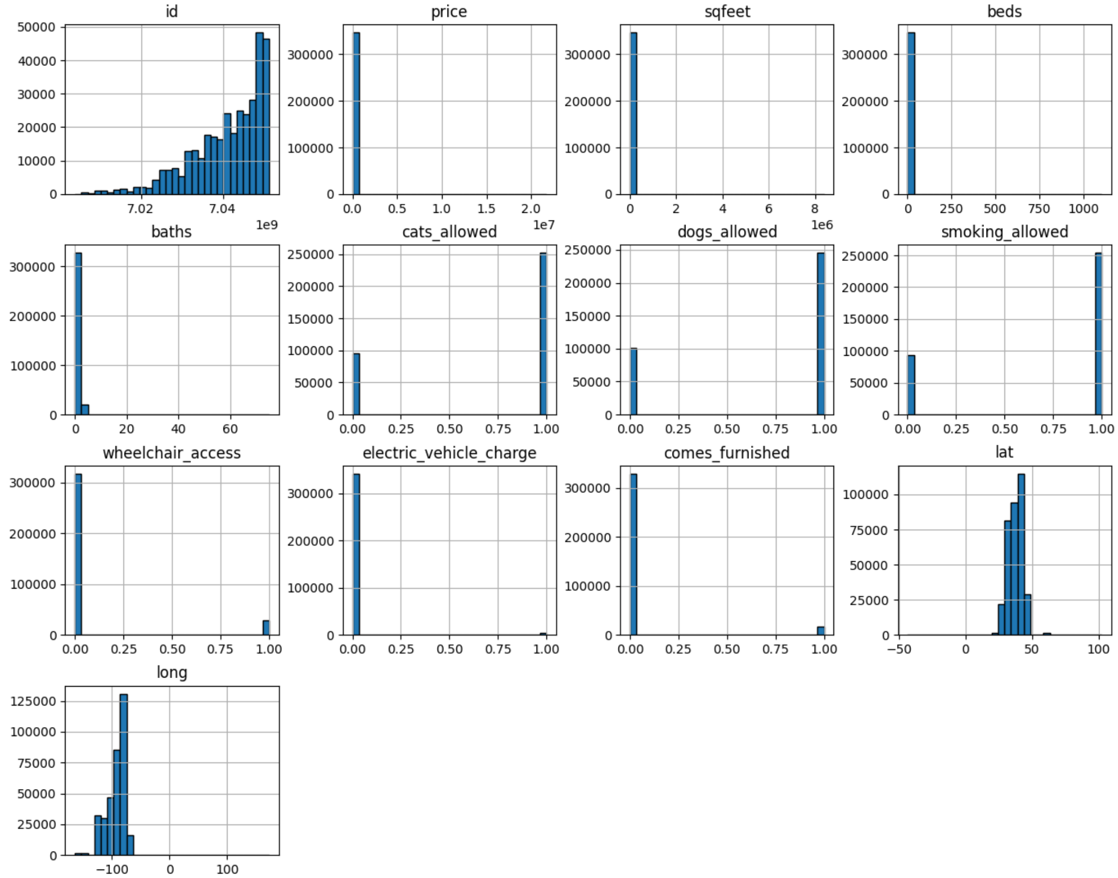
Cleaned Word Cloud of Description Text


```
[29]: # pd.set_option('display.max_colwidth', None)
      # print(train_df['description'][100:110])
      # pd.reset_option('display.max_colwidth')
```

0.2.9 Histogram of numerical features for comparison

```
[30]: train_df.hist(figsize=(15, 12), bins=30, edgecolor="black")
```

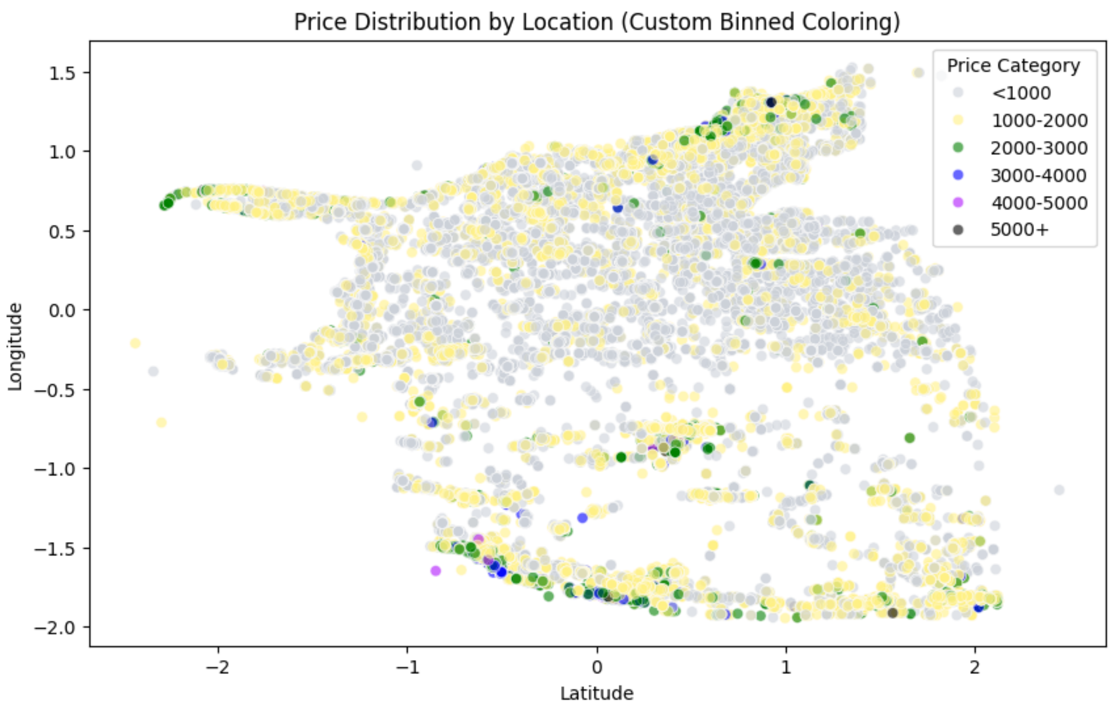
```
[30]: array([[<Axes: title={'center': 'id'}>,
              <Axes: title={'center': 'price'}>,
              <Axes: title={'center': 'sqfeet'}>,
              <Axes: title={'center': 'beds'}>],
             [<Axes: title={'center': 'baths'}>,
              <Axes: title={'center': 'cats_allowed'}>,
              <Axes: title={'center': 'dogs_allowed'}>,
              <Axes: title={'center': 'smoking_allowed'}>],
             [<Axes: title={'center': 'wheelchair_access'}>,
              <Axes: title={'center': 'electric_vehicle_charge'}>,
              <Axes: title={'center': 'comes_furnished'}>,
              <Axes: title={'center': 'lat'}>],
             [<Axes: title={'center': 'long'}>, <Axes: >, <Axes: >, <Axes: >]],
            dtype=object)
```



Key Takeaway: - Price and sqfeet are skewed, normal and even distribution is desired, which we achieve after removing outlier and log base it

[174]:
```
# Define price bins and corresponding colors
# price_bins = [0, 1000, 2000, 3000, 4000, 5000, np.inf]
# price_labels = ["<1000", "1000-2000", "2000-3000", "3000-4000", "4000-5000",␣
↪"5000+"]

# # Create a new column for price categories
# train_df["price_category"] = pd.cut(train_df["price"], bins=price_bins,␣
↪labels=price_labels, include_lowest=True)

# # Filter dataset based on latitude and longitude thresholds
# filtered_df = train_df[(train_df["lat"] >= -2.5) & (train_df["lat"] <= 2.5) &
#                          (train_df["long"] >= -2.5) & (train_df["long"] <= 2.
↪5)]

# # Scatter plot with custom price categories
# plt.figure(figsize=(10, 6))
```

```
# sns.scatterplot(x="lat", y="long", data=filtered_df, hue="price_category",␣
↪palette={"<1000": "#ccd1d9", "1000-2000": "#fff185", "2000-3000": "green",␣
↪"3000-4000": "blue", "4000-5000": "#af14fc", "5000+": "black"}, alpha=0.6)

# # Labels and title
# plt.xlabel("Latitude")
# plt.ylabel("Longitude")
# plt.title("Price Distribution by Location (Custom Binned Coloring)")
# plt.legend(title="Price Category", loc="upper right")
# plt.show()
```



0.3 Data Preprocessing

0.3.1 2. Handling Missing Values

0.3.2 2.1 Removing Outlier in prices

Reason: - Mentioned above in the Box-plot - We can see from the price min and max values that most of the prices are in 0, 10^3 range. But, outliers are far away (10^7)

[31]:
```
train_df.shape
```

[31]: (346479, 19)

[32]:
```
# Filtering properties using IQR
# Apply log transformation to normalize price
train_df["log_price"] = np.log1p(train_df["price"])

# Compute IQR on log-transformed prices
Q1_log = train_df["log_price"].quantile(0.25)
Q3_log = train_df["log_price"].quantile(0.75)
IQR_log = Q3_log - Q1_log

# Define new bounds using IQR (less strict)
lower_bound_log = Q1_log - (3 * IQR_log)
upper_bound_log = Q3_log + (3 * IQR_log)

# Convert bounds back to original scale
lower_bound = np.expm1(lower_bound_log)
upper_bound = np.expm1(upper_bound_log)

print('price range: ', lower_bound, upper_bound)

# Filter price using log-adjusted bounds
train_df = train_df[(train_df["price"] >= lower_bound) & (train_df["price"] <=
 ↪upper_bound)]

# Drop the temporary log-transformed price column after filtering
train_df.drop(columns=["log_price"], inplace=True)
```

```
price range:  154.89704526295714 7225.384554625592
```

[33]:
```
train_df.shape
```

[33]:
```
(344060, 19)
```

[34]:
```
# Removing zero price properties
train_df = train_df[(train_df["price"] > 0)]
```

[35]:
```
train_df.shape
```

[35]:
```
(344060, 19)
```

[36]:
```
# test
test_df["log_price"] = np.log1p(test_df["price"])

Q1_log = test_df["log_price"].quantile(0.25)
Q3_log = test_df["log_price"].quantile(0.75)
IQR_log = Q3_log - Q1_log

lower_bound_log = Q1_log - (3 * IQR_log)
upper_bound_log = Q3_log + (3 * IQR_log)
```

```
lower_bound = np.expm1(lower_bound_log)
upper_bound = np.expm1(upper_bound_log)

print('price range: ', lower_bound, upper_bound)

test_df = test_df[(test_df["price"] >= lower_bound) & (test_df["price"] <=
 ↪upper_bound)]
test_df.drop(columns=["log_price"], inplace=True)
```

```
price range:  154.12575647430336 7252.3151526412485
```

0.3.3 2.2 Removing Outlier in sqfeet

- Drop zero sqfeet enteries
- Drop extreme properties

Log values are used so tackle high skewness

[37]:
```
# Remove rows where sqfeet is 0
train_df = train_df[train_df["sqfeet"] > 0]
```

[38]:
```
test_df = test_df[test_df["sqfeet"] > 0]
```

[39]:
```
train_df.shape
```

[39]:
```
(344014, 19)
```

[40]:
```
# Removing IQR outside range for sqfeet
# Apply log transformation to normalize sqfeet due to skewness
train_df["log_sqfeet"] = np.log1p(train_df["sqfeet"])

# Compute IQR on log-transformed prices
Q1_sqfeet = train_df["log_sqfeet"].quantile(0.25)
Q3_sqfeet = train_df["log_sqfeet"].quantile(0.75)
IQR_sqfeet = Q3_sqfeet - Q1_sqfeet

# Define new bounds using IQR (less strict)
lower_bound_sqfeet = Q1_sqfeet - (3 * IQR_sqfeet)
upper_bound_sqfeet = Q3_sqfeet + (3 * IQR_sqfeet)

# Convert bounds back to original scale
lower_bound_sqfeet = np.expm1(lower_bound_sqfeet)
upper_bound_sqfeet = np.expm1(upper_bound_sqfeet)

print('sqfeet range: ', lower_bound_sqfeet, upper_bound_sqfeet)

# Filter price using log-adjusted bounds
```

```
train_df = train_df[(train_df["sqfeet"] >= lower_bound_sqfeet) &␣
 ↪(train_df["sqfeet"] <= upper_bound_sqfeet)]

# Drop the temporary log-transformed price column after filtering
train_df.drop(columns=["log_sqfeet"], inplace=True)

# Q1_sqfeet = train_df["sqfeet"].quantile(0.25)
# Q3_sqfeet = train_df["sqfeet"].quantile(0.75)
# IQR_sqfeet = Q3_sqfeet - Q1_sqfeet
# train_df = train_df[(train_df["sqfeet"] >= (Q1_sqfeet - 1.5 * IQR_sqfeet)) &␣
 ↪(train_df["sqfeet"] <= (Q3_sqfeet + 1.5 * IQR_sqfeet))]
# sqfeet range: 150.0 1750.0
```

```
sqfeet range:  207.6093534841274 4142.634911680846
```

[41]:
```
# test
test_df["log_sqfeet"] = np.log1p(test_df["sqfeet"])

Q1_sqfeet = test_df["log_sqfeet"].quantile(0.25)
Q3_sqfeet = test_df["log_sqfeet"].quantile(0.75)
IQR_sqfeet = Q3_sqfeet - Q1_sqfeet

lower_bound_sqfeet = Q1_sqfeet - (3 * IQR_sqfeet)
upper_bound_sqfeet = Q3_sqfeet + (3 * IQR_sqfeet)

lower_bound_sqfeet = np.expm1(lower_bound_sqfeet)
upper_bound_sqfeet = np.expm1(upper_bound_sqfeet)

test_df = test_df[(test_df["sqfeet"] >= lower_bound_sqfeet) &␣
 ↪(test_df["sqfeet"] <= upper_bound_sqfeet)]

# Drop the temporary log-transformed price column after filtering
test_df.drop(columns=["log_sqfeet"], inplace=True)
```

[42]:
```
train_df.shape
```

[42]:
```
(342638, 19)
```

0.3.4 2.3 Removing Outliers in beds and baths

- 0 bed properties are outliers
- Properties with 1000+ beds and 25+ baths are outliers

[43]:
```
# Removing 0 bed properties
train_df = train_df[(train_df["beds"] > 0)]
test_df = test_df[(test_df["beds"] > 0)]
```

[44]:
```
train_df.shape
```

[44]: (332944, 19)

[45]:
```python
## Remove outliers in beds
train_df = train_df[(train_df["beds"] < 1000)]
test_df = test_df[(test_df["beds"] < 1000)]
```

[46]:
```python
train_df.shape
```

[46]: (332941, 19)

[47]:
```python
## Remove properties with unrealistic baths
train_df = train_df[train_df["baths"] < 25]
test_df = test_df[test_df["baths"] < 25]
```

[48]:
```python
train_df.shape
```

[48]: (332941, 19)

0.3.5 2.4 Drop missing Description

Reasoning: Only 2 rows were missing description out of 346,479, which is 0.0006% of the data. Removing them does not significantly impact the dataset.

[49]:
```python
train_df.dropna(subset=["description"], inplace=True)
test_df.dropna(subset=["description"], inplace=True)
```

[50]:
```python
train_df.shape
```

[50]: (332939, 19)

0.3.6 2.5 Filling Lat and Lng Missing Values

Reasoning: Ensuresuing missing values are filled with the median specific to their region, improving accuracy.

[51]:
```python
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 332939 entries, 0 to 346478
Data columns (total 19 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   id                       332939 non-null  int64
 1   region                   332939 non-null  object
 2   price                    332939 non-null  int64
 3   type                     332939 non-null  object
 4   sqfeet                   332939 non-null  int64
 5   beds                     332939 non-null  int64
 6   baths                    332939 non-null  float64
```

```
 7   cats_allowed             332939 non-null  int64
 8   dogs_allowed             332939 non-null  int64
 9   smoking_allowed          332939 non-null  int64
 10  wheelchair_access        332939 non-null  int64
 11  electric_vehicle_charge  332939 non-null  int64
 12  comes_furnished          332939 non-null  int64
 13  laundry_options          263927 non-null  object
 14  parking_options          210920 non-null  object
 15  description              332939 non-null  object
 16  lat                      331258 non-null  float64
 17  long                     331258 non-null  float64
 18  state                    332939 non-null  object
dtypes: float64(3), int64(10), object(6)
memory usage: 50.8+ MB
```

[52]:
```python
train_df["lat"] = train_df.groupby("region")["lat"].transform(lambda x: x.
 ↪fillna(x.median()))
train_df["long"] = train_df.groupby("region")["long"].transform(lambda x: x.
 ↪fillna(x.median()))

test_df["lat"] = test_df.groupby("region")["lat"].transform(lambda x: x.
 ↪fillna(x.median()))
test_df["long"] = test_df.groupby("region")["long"].transform(lambda x: x.
 ↪fillna(x.median()))
```

[53]:
```python
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 332939 entries, 0 to 346478
Data columns (total 19 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   id                       332939 non-null  int64
 1   region                   332939 non-null  object
 2   price                    332939 non-null  int64
 3   type                     332939 non-null  object
 4   sqfeet                   332939 non-null  int64
 5   beds                     332939 non-null  int64
 6   baths                    332939 non-null  float64
 7   cats_allowed             332939 non-null  int64
 8   dogs_allowed             332939 non-null  int64
 9   smoking_allowed          332939 non-null  int64
 10  wheelchair_access        332939 non-null  int64
 11  electric_vehicle_charge  332939 non-null  int64
 12  comes_furnished          332939 non-null  int64
 13  laundry_options          263927 non-null  object
 14  parking_options          210920 non-null  object
 15  description              332939 non-null  object
```

```
 16  lat                      332939 non-null  float64
 17  long                     332939 non-null  float64
 18  state                    332939 non-null  object
dtypes: float64(3), int64(10), object(6)
memory usage: 50.8+ MB
```

0.3.7 2.6 Filling Missing Categorical Values

- Laundary Options
- Parking options

Reason: Filling missing values with the mode ensures that the most realistic and frequently used option is assigned.

[54]:
```python
# train_df["laundry_options"].fillna(train_df["laundry_options"].mode()[0], inplace=True)
# train_df["parking_options"].fillna(train_df["parking_options"].mode()[0], inplace=True)
train_df["laundry_options"] = train_df["laundry_options"].fillna(train_df["laundry_options"].mode()[0])
train_df["parking_options"] = train_df["parking_options"].fillna(train_df["parking_options"].mode()[0])

test_df["laundry_options"] = test_df["laundry_options"].fillna(test_df["laundry_options"].mode()[0])
test_df["parking_options"] = test_df["parking_options"].fillna(test_df["parking_options"].mode()[0])
```

[55]:
```python
print(train_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 332939 entries, 0 to 346478
Data columns (total 19 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   id                       332939 non-null  int64
 1   region                   332939 non-null  object
 2   price                    332939 non-null  int64
 3   type                     332939 non-null  object
 4   sqfeet                   332939 non-null  int64
 5   beds                     332939 non-null  int64
 6   baths                    332939 non-null  float64
 7   cats_allowed             332939 non-null  int64
 8   dogs_allowed             332939 non-null  int64
 9   smoking_allowed          332939 non-null  int64
 10  wheelchair_access        332939 non-null  int64
 11  electric_vehicle_charge  332939 non-null  int64
 12  comes_furnished          332939 non-null  int64
 13  laundry_options          332939 non-null  object
```

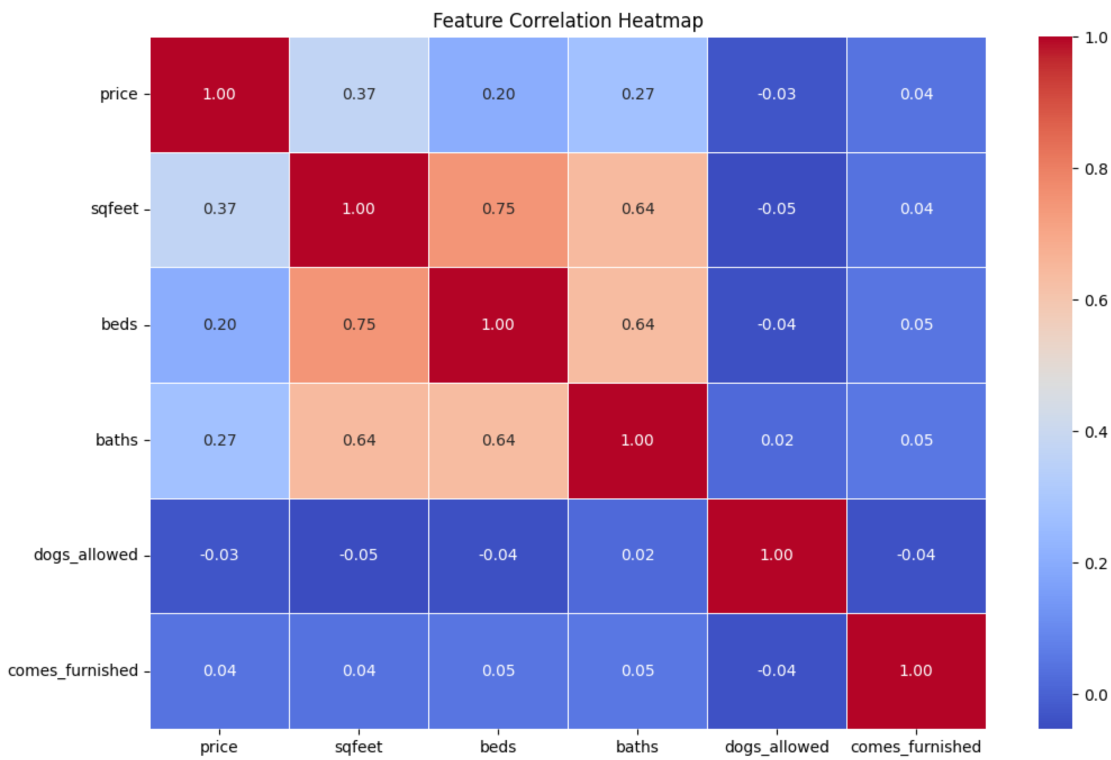
```
 14  parking_options          332939 non-null  object
 15  description              332939 non-null  object
 16  lat                      332939 non-null  float64
 17  long                     332939 non-null  float64
 18  state                    332939 non-null  object
dtypes: float64(3), int64(10), object(6)
memory usage: 50.8+ MB
None
```

[56]:
```
train_df.describe()
```

[56]:
```
                 id          price         sqfeet           beds  \
count  3.329390e+05  332939.000000  332939.000000  332939.000000
mean   7.040948e+09    1190.087316    1007.999162       1.947990
std    8.808727e+06     574.643847     363.995150       0.823293
min    7.003808e+09     157.000000     210.000000       1.000000
25%    7.035952e+09     820.000000     766.000000       1.000000
50%    7.043143e+09    1049.000000     950.000000       2.000000
75%    7.048411e+09    1399.000000    1153.000000       2.000000
max    7.051292e+09    7200.000000    4136.000000       8.000000

               baths   cats_allowed   dogs_allowed  smoking_allowed  \
count  332939.000000  332939.000000  332939.000000    332939.000000
mean        1.492473       0.727542       0.709310         0.735910
std         0.586911       0.445225       0.454081         0.440848
min         0.000000       0.000000       0.000000         0.000000
25%         1.000000       0.000000       0.000000         0.000000
50%         1.000000       1.000000       1.000000         1.000000
75%         2.000000       1.000000       1.000000         1.000000
max         7.500000       1.000000       1.000000         1.000000

       wheelchair_access  electric_vehicle_charge  comes_furnished  \
count      332939.000000            332939.000000    332939.000000
mean            0.079210                 0.012537         0.043963
std             0.270066                 0.111264         0.205013
min             0.000000                 0.000000         0.000000
25%             0.000000                 0.000000         0.000000
50%             0.000000                 0.000000         0.000000
75%             0.000000                 0.000000         0.000000
max             1.000000                 1.000000         1.000000

                 lat           long
count  332939.000000  332939.000000
mean       37.216456     -92.530496
std         5.546563      16.431466
min       -43.533300    -163.894000
25%        33.422600     -98.611700
```

```
50%        37.623600     -87.625700
75%        41.137900     -81.162800
max       102.036000     172.633000
```

[57]:
```python
selected_cols = ["price", "sqfeet", "beds", "baths", "dogs_allowed",␣
↪"comes_furnished"]

# Compute the correlation matrix
corr_matrix = train_df[selected_cols].corr()

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```

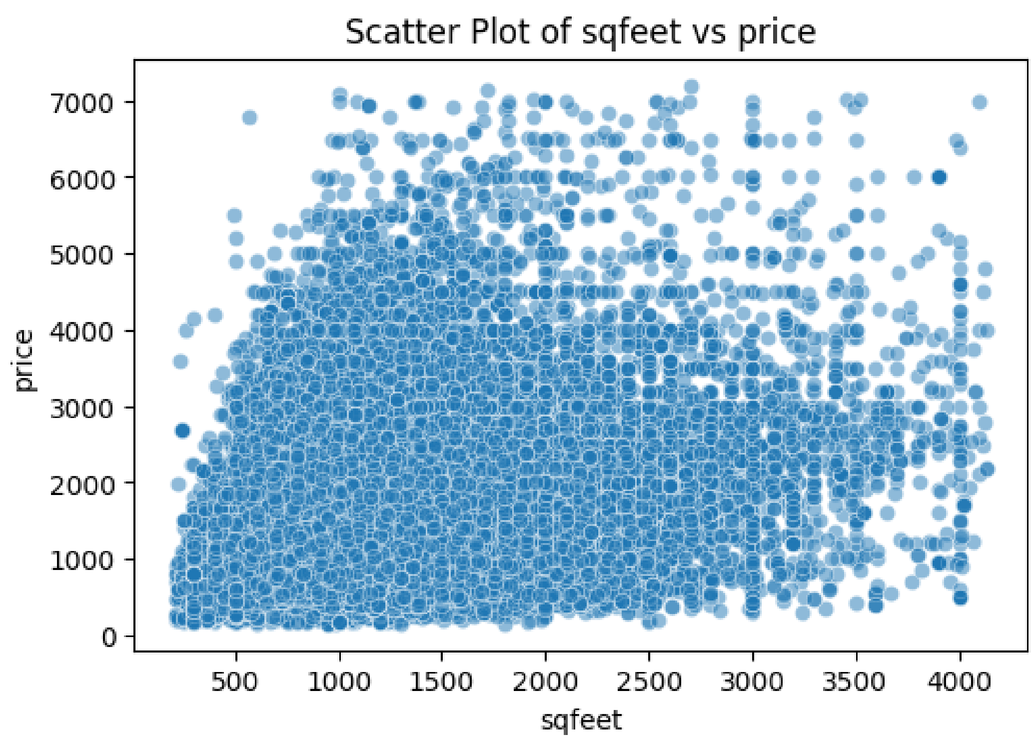


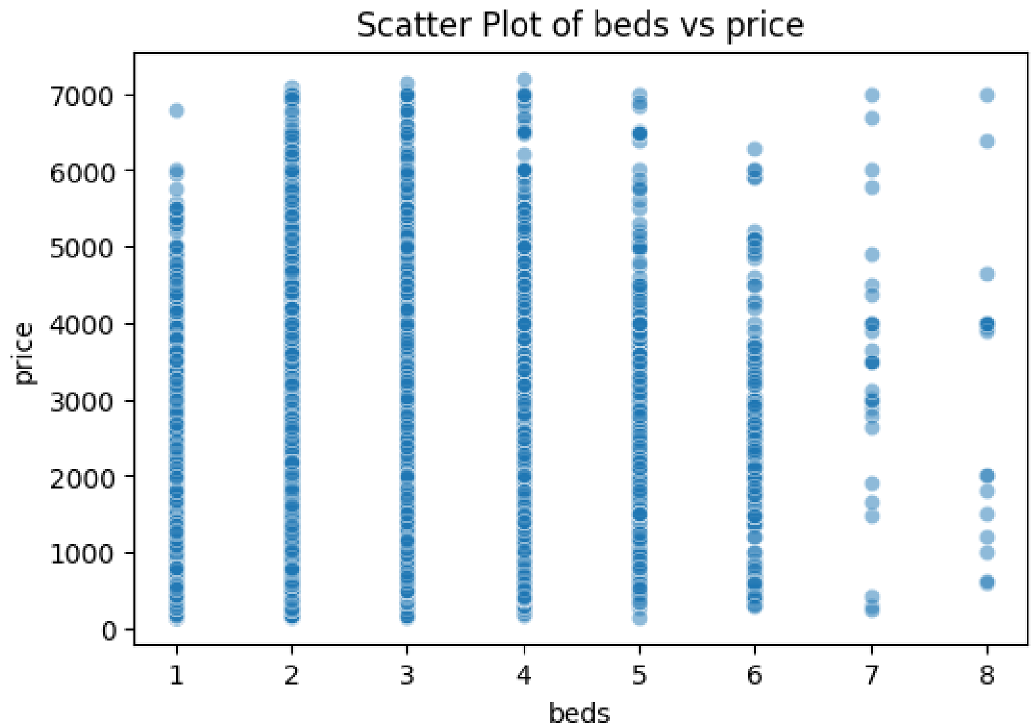
Key Insights - Square footage is the most important numerical feature for price prediction. - Bedrooms and bathrooms have weak correlations, meaning size matters more than room count. - Pet-friendliness and furnishing don't significantly impact rent.

- Beds, baths, and sqfeet are highly correlated, meaning **multicollinearity** exists.
- We might remove beds or baths in some models (to avoid redundancy).
  - Predicting price using both beds and sqfeet might not add extra value.

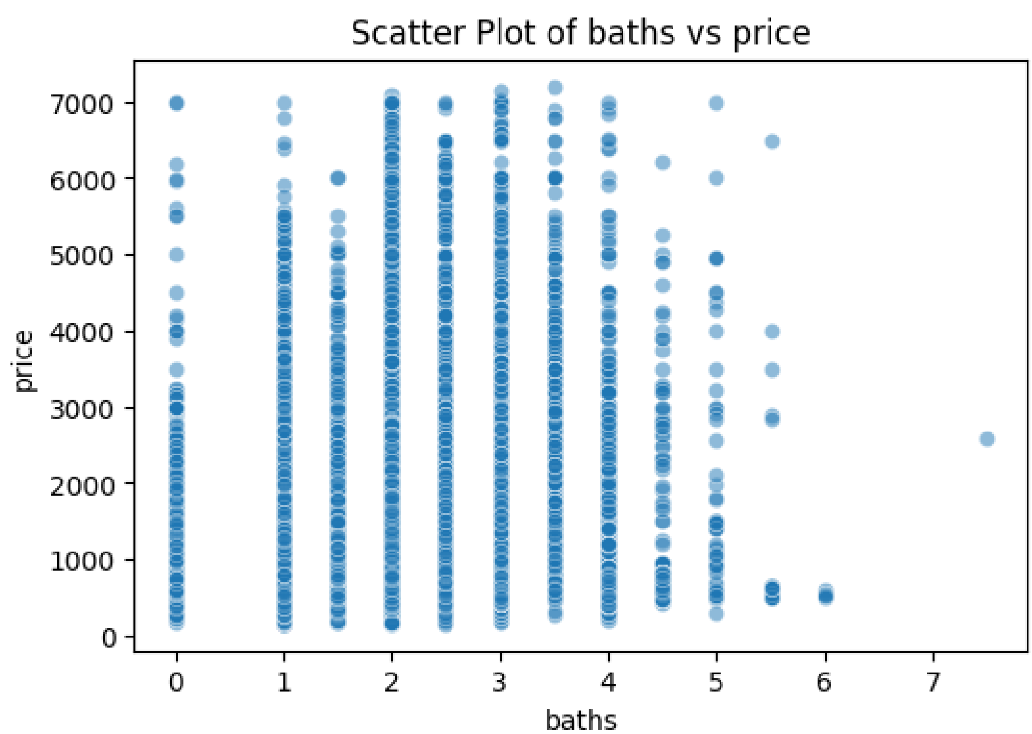
[58]:
```
def plot_scatter(df, columns, target="price"):
    plt.figure(figsize=(12, 8))
    for col in columns:
        plt.figure(figsize=(6, 4))
        sns.scatterplot(x=df[col], y=df[target], alpha=0.5)
        plt.xlabel(col)
        plt.ylabel(target)
        plt.title(f"Scatter Plot of {col} vs {target}")
        plt.show()

numerical_cols = ["sqfeet", "beds", "baths", "lat", "long"]
plot_scatter(train_df, numerical_cols, target="price")
```

```
<Figure size 1200x800 with 0 Axes>
```

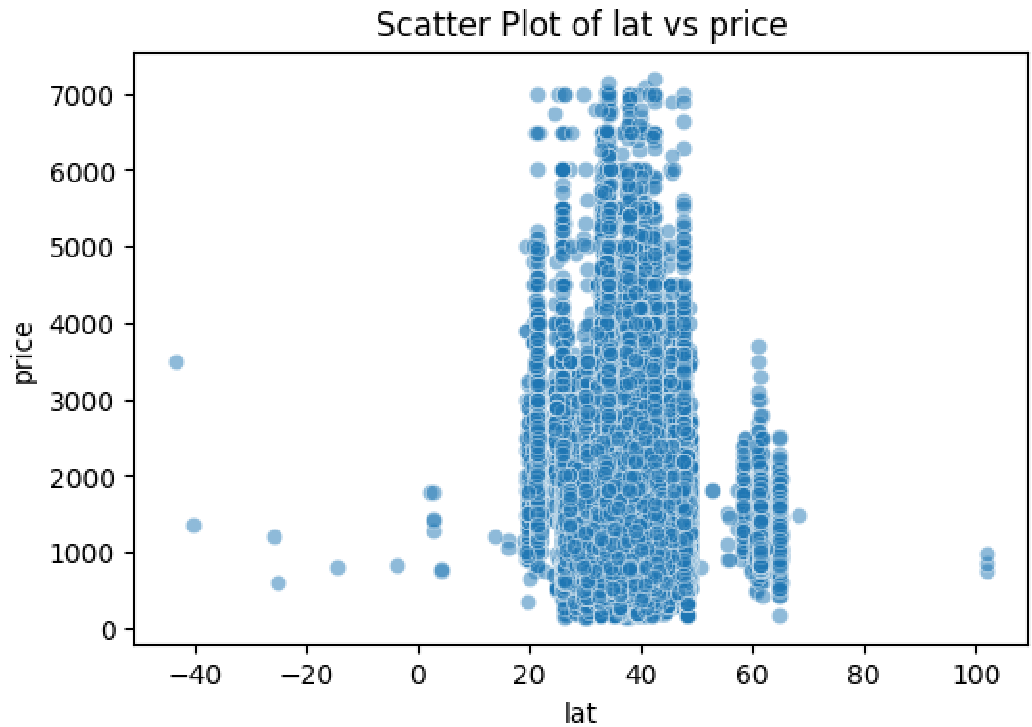


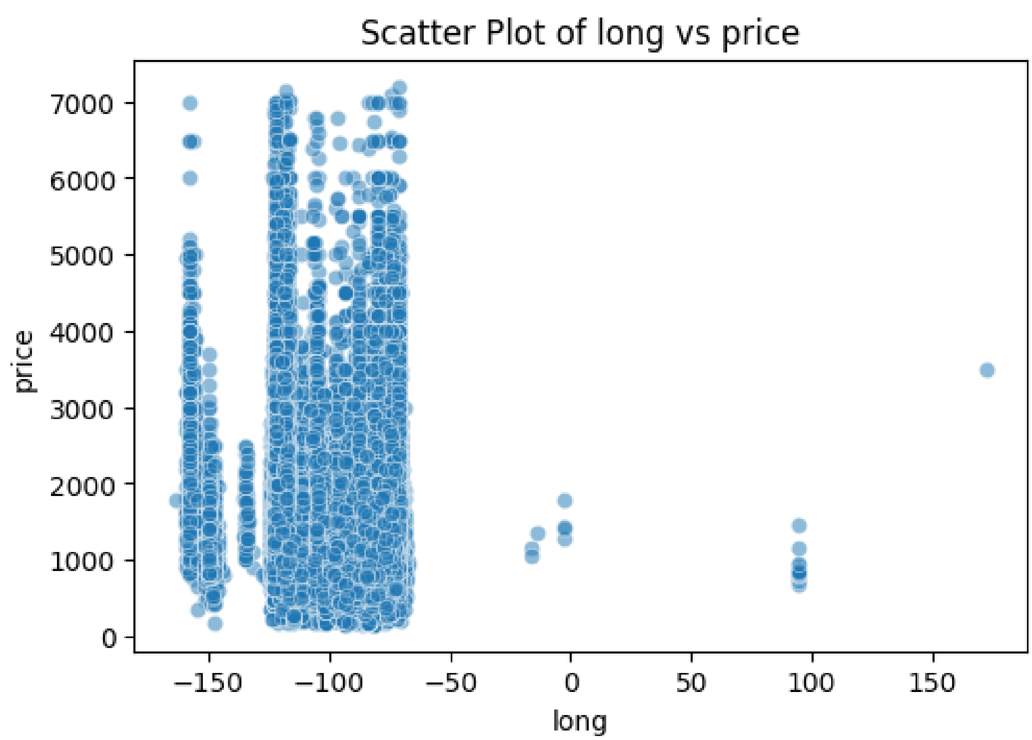


Scatter Plot of beds vs price
price
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
beds



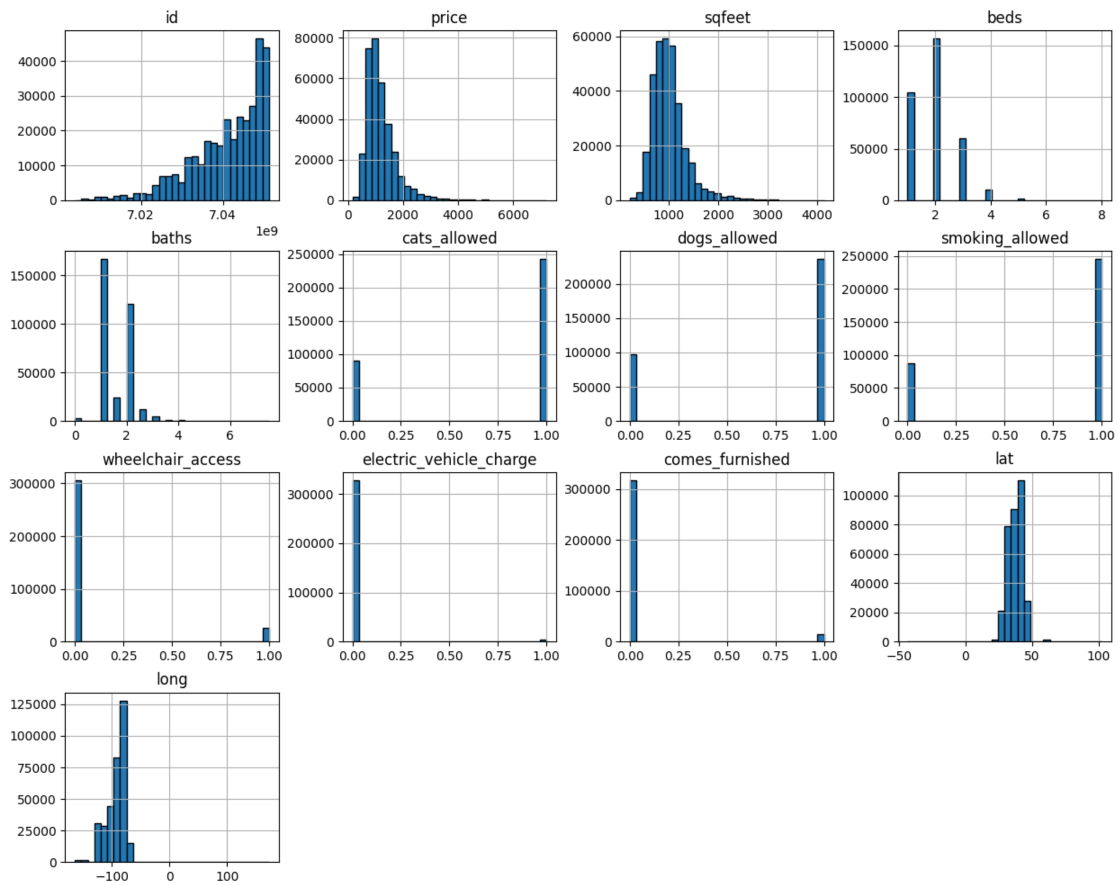
Scatter Plot of baths vs price
price
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
6
7
baths


Scatter Plot of lat vs price
price
lat
7000
6000
5000
4000
3000
2000
1000
0
−40
−20
0
20
40
60
80
100


Scatter Plot of long vs price
price
long
7000
6000
5000
4000
3000
2000
1000
0
−150
−100
−50
0
50
100
150

[59]:
```python
train_df.hist(figsize=(15, 12), bins=30, edgecolor="black")
```

[59]:
```
array([[<Axes: title={'center': 'id'}>,
        <Axes: title={'center': 'price'}>,
        <Axes: title={'center': 'sqfeet'}>,
        <Axes: title={'center': 'beds'}>],
       [<Axes: title={'center': 'baths'}>,
        <Axes: title={'center': 'cats_allowed'}>,
        <Axes: title={'center': 'dogs_allowed'}>,
        <Axes: title={'center': 'smoking_allowed'}>],
       [<Axes: title={'center': 'wheelchair_access'}>,
        <Axes: title={'center': 'electric_vehicle_charge'}>,
        <Axes: title={'center': 'comes_furnished'}>,
        <Axes: title={'center': 'lat'}>],
       [<Axes: title={'center': 'long'}>, <Axes: >, <Axes: >, <Axes: >]],
      dtype=object)
```



0.4 Feature Engineering

0.4.1 3.1 Creating new features

[60]:
```
# Price per sq
# Not included because of leaking issue (we want to predict price)
train_df["price_per_sqft"] = train_df["price"] / train_df["sqfeet"]
test_df["price_per_sqft"] = test_df["price"] / test_df["sqfeet"]
```

[61]:
```
# Total pets allowed
train_df["total_pets"] = train_df["cats_allowed"] + train_df["dogs_allowed"]
test_df["total_pets"] = test_df["cats_allowed"] + test_df["dogs_allowed"]
```

[62]:
```
# Luxury feature allowed (Wheelchair, electric, comes_furnihed)
train_df["luxury_features"] = (train_df["electric_vehicle_charge"] +␣
↪train_df["wheelchair_access"] + train_df["comes_furnished"])
test_df["luxury_features"] = (test_df["electric_vehicle_charge"] +␣
↪test_df["wheelchair_access"] + test_df["comes_furnished"])
```

[63]:
```
# 4. Location cluster feature based on Lat and Lng
# 12 general regions in US
coords = train_df[['lat', 'long']]
kmeans = KMeans(n_clusters=10, random_state=42)
train_df['geo_cluster'] = kmeans.fit_predict(coords)

coords = test_df[['lat', 'long']]
kmeans = KMeans(n_clusters=10, random_state=42)
test_df['geo_cluster'] = kmeans.fit_predict(coords)
```

[64]:
```
# Frequency encoding for region
region_freq = train_df['region'].value_counts(normalize=True)
train_df['region_freq'] = train_df['region'].map(region_freq)

region_freq = test_df['region'].value_counts(normalize=True)
test_df['region_freq'] = test_df['region'].map(region_freq)
```

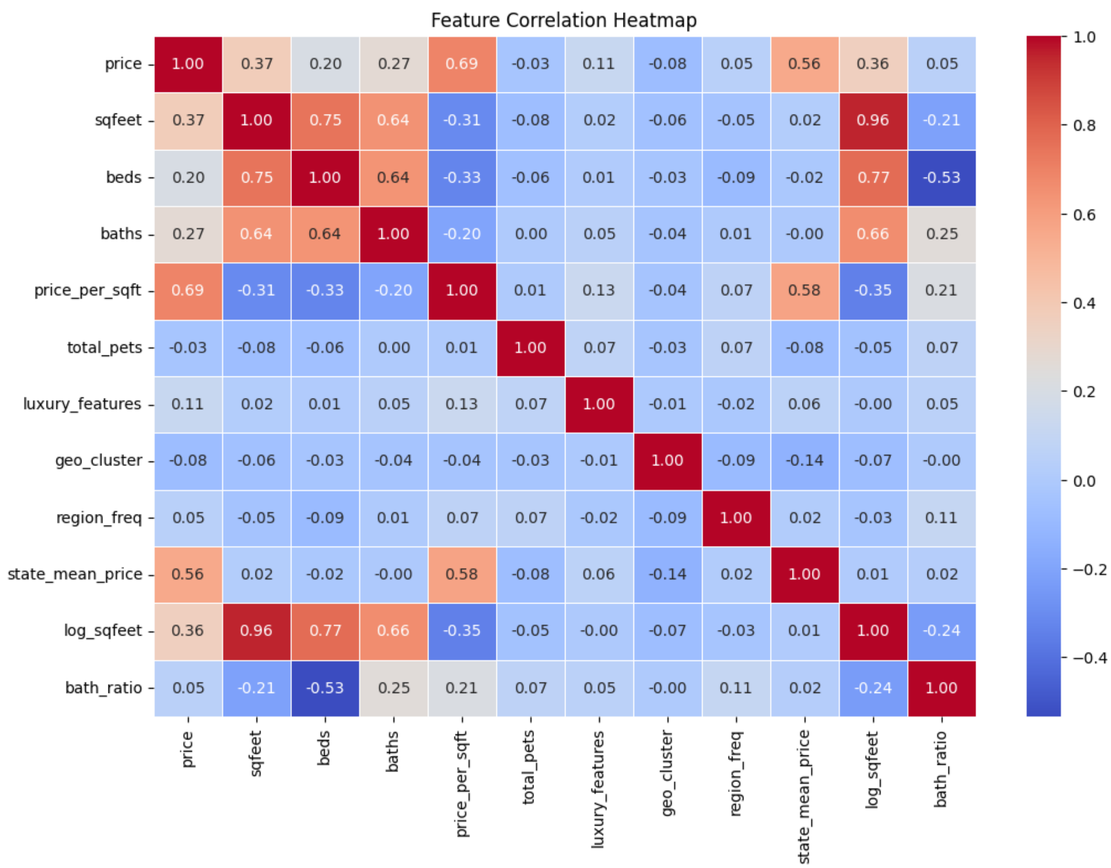
[65]:
```
# Apply log transformation to skewed features
train_df["log_price"] = np.log1p(train_df["price"])
train_df["log_sqfeet"] = np.log1p(train_df["sqfeet"])

test_df["log_price"] = np.log1p(test_df["price"])
test_df["log_sqfeet"] = np.log1p(test_df["sqfeet"])
```

[66]:
```
# Target encoding for state
state_mean_price = train_df.groupby('state')['price'].mean()
train_df['state_mean_price'] = train_df['state'].map(state_mean_price)

state_mean_price = test_df.groupby('state')['price'].mean()
```

```
test_df['state_mean_price'] = test_df['state'].map(state_mean_price)
```

[67]:
```
# Bath ratio
train_df['bath_ratio'] = train_df['baths'] / train_df['beds']

test_df['bath_ratio'] = test_df['baths'] / test_df['beds']
```

[68]:
```
selected_cols = ["price", "sqfeet", "beds", "baths", "price_per_sqft",␣
↪"total_pets", "luxury_features", "geo_cluster", "region_freq",␣
↪"state_mean_price", "log_sqfeet", "bath_ratio"]

# Compute the correlation matrix
corr_matrix = train_df[selected_cols].corr()

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```



[69]:
```
# Dropping price_per_sqfeet due to unfair advantage in training and prediction
train_df.drop(columns=["price_per_sqft"], inplace=True)
test_df.drop(columns=["price_per_sqft"], inplace=True)
```

[70]:
```
train_df.drop(columns=["state_mean_price"], inplace=True)
test_df.drop(columns=["state_mean_price"], inplace=True)
```

0.4.2 3.2 Text feature engineering

0.4.3 Description to numerical values using TF-IDF vectorization

Other Options - Use BERT embedding (768 additional column for meaningful description, PCA/Autoencoders to reduce it to 100 again) - Using Spacy NER, we can have meaningful additional columns that could impact the price, but they were taking too much time (2+ hrs)

0.4.4 NER

[71]:
```
# def preprocess_text(text):
#     text = text.lower()
#     text = re.sub(r'[^\w\s]', '', text)
#     return text

# train_df['description_cleaned'] = train_df['description'].
↪apply(preprocess_text)
```

[72]:
```
# vectorizer = TfidfVectorizer(stop_words="english", max_features=100,␣
↪ngram_range=(1, 2))
# desc_features = vectorizer.fit_transform(train_df["description_cleaned"])
# # Convert to DataFrame and merge
# desc_df = pd.DataFrame(desc_features.toarray(), columns=[f"desc_{i}" for i in␣
↪range(100)])
```

[73]:
```
# train_df = pd.concat([train_df.reset_index(drop=True), desc_df], axis=1)
# # Drop raw description column
# train_df.drop(columns=["description"], inplace=True)
```

[74]:
```
# Load spaCy's pre-trained English model
# nlp = spacy.load("en_core_web_sm", disable=["tagger", "parser"])
```

[75]:
```
# Define the entity labels we care about
# relevant_entities = {"FAC", "ORG", "GPE", "LOC", "MONEY"}  # Only these␣
↪entities

# Function to extract binary entity flags (FAST)
# def extract_entity_flags(texts):
#     entity_flags = []
```

```
#     for doc in tqdm(nlp.pipe(texts, batch_size=500), total=len(texts),␣
↪desc="Processing NER"):
#         entities_present = {label: 0 for label in relevant_entities}  #␣
↪Initialize all to 0
#         for ent in doc.ents:
#             if ent.label_ in relevant_entities:
#                 entities_present[ent.label_] = 1  # Mark as present
#         entity_flags.append(entities_present)

#     return entity_flags
```

[76]:
```
# train_df.drop(columns=["named_entities"], inplace=True)
# test_df.drop(columns=["named_entities"], inplace=True)
```

[77]:
```
# Convert descriptions to strings (avoid NaN issues)
# train_df['description'] = train_df['description'].astype(str)
# test_df['description'] = test_df['description'].astype(str)
```

[78]:
```
# Apply batch processing with progress tracking
# train_entity_flags = extract_entity_flags(train_df['description'])
# test_entity_flags = extract_entity_flags(test_df['description'])
```

[79]:
```
# Convert results to DataFrame and merge
# train_entity_df = pd.DataFrame(train_entity_flags)
# test_entity_df = pd.DataFrame(test_entity_flags)

# train_df = pd.concat([train_df, train_entity_df], axis=1)
# test_df = pd.concat([test_df, test_entity_df], axis=1)
```

[80]:
```
# pd.set_option('display.max_colwidth', None)
# print(train_df['description'][100:110])
# pd.reset_option('display.max_colwidth')
# train_df.head(10)[['named_entities', 'description']]
```

[81]:
```
# train_df.head()['named_entities']
```

[82]:
```
train_df.shape
```

[82]:
```
(332939, 26)
```

0.4.5 Keyword-based Binary Flags

[83]:
```
feature_keywords = {
    "has_outdoor": ["tennis", "court", "business center", "park", "community",␣
↪"clubhouse"],
    "has_pool": ["pool", "swimming"],
    "has_fitness_center": ["gym", "fitness center"],
```

```
    "mentions_price": ["rent", "deposit", "fee", "monthly"],
    "has_maintenance": ["emergency", "maintenance", "onsite"],
    "has_management": ["management", "managed", "professionally", "office"],
    "has_garbage_disposal": ["garbage", "disposal"],
    "has_closet": ["closet", "closets", "storage"],
    "has_kitchen": ["dining", "kitchen", "refrigerator"],
    "has_fan": ["fan", "air"],
    "has_offer": ["offer", "offers", "discount"],
    "has_appliances": ["appliances", "dryer", "laundry", "washer"]
}

# 'access', 'air', 'amenities', 'apartment', 'apartments',
# 'appliances', 'application', 'apply', 'area', 'availability',
# 'available', 'bath', 'beautiful', 'bedroom', 'bedrooms', 'ceiling',
# 'center', 'central', 'change', 'close', 'closet', 'closets',
# 'clubhouse', 'come', 'community', 'contact', 'court', 'deposit',
# 'details', 'dining', 'dishwasher', 'dogs', 'downtown', 'dryer',
# 'easy', 'enjoy', 'equal', 'features', 'fee', 'fitness', 'floor',
# 'flooring', 'free', 'friendly', 'ft', 'great', 'home', 'homes',
# 'housing', 'included', 'info', 'just', 'kitchen', 'large',
# 'laundry', 'lease', 'leasing', 'living', 'located', 'location',
# 'maintenance', 'management', 'minutes', 'month', 'new', 'offer',
# 'offers', 'office', 'online', 'onsite', 'opportunity', 'park',
# 'parking', 'patio', 'pet', 'pets', 'plans', 'pool', 'private',
# 'property', 'refrigerator', 'rent', 'room', 'schedule', 'select',
# 'shopping', 'space', 'spacious', 'sq', 'storage', 'subject',
# 'swimming', 'today', 'tour', 'unit', 'units', 'walkin', 'washer',
# 'water', 'welcome'

# Function to check for keyword presence
def keyword_flag(text, keywords):
    return bool(any(word in str(text).lower() for word in keywords))
```

[84]:
```
# Apply to dataset
for feature, words in feature_keywords.items():
    train_df[feature] = train_df['description'].apply(lambda x: keyword_flag(x,␣
↪words))
```

[85]:
```
train_df.shape
```

[85]:
```
(332939, 38)
```

[86]:
```
train_df.head()
```

[86]:
```
           id           region  price       type  sqfeet  beds  baths  \
0  7049965813    orange county   2632  apartment    1080     2    2.0
1  7036046796   visalia-tulare   1160  apartment     768     2    1.0
```

```
2  7037856890             portland   1262  apartment    1075     2    1.0
3  7046933042              boulder   1861  apartment    1076     2    2.0
4  7048650961  sioux falls / SE SD    626  apartment     720     1    1.0

   cats_allowed  dogs_allowed  smoking_allowed  …  has_fitness_center  \
0             1             1                0  …                True
1             1             1                1  …               False
2             1             1                0  …               False
3             1             1                0  …               False
4             1             1                0  …                True

   mentions_price  has_maintenance has_management has_garbage_disposal  \
0            True            False          False                 True
1           False            False           True                False
2           False             True           True                False
3            True             True           True                False
4           False            False          False                False

  has_closet  has_kitchen  has_fan has_offer  has_appliances
0      False        False     True      True            True
1      False        False    False     False           False
2      False        False    False      True            True
3       True         True     True      True            True
4      False        False    False     False           False

[5 rows x 38 columns]
```

[87]:
```
for feature, words in feature_keywords.items():
    test_df[feature] = test_df['description'].apply(lambda x: keyword_flag(x,␣
↪words))
```

[88]:
```
test_df.shape
```

[88]:
```
(36947, 38)
```

0.4.6 4. Encode Categorical Variables

0.4.7 4.1 One-Hot Encoding for Nominal Categories

For model to train, convert `region`, `type`, … to numerical values

- TODO: Use clustering to encode region column

0.4.8 Type, Laundary, Parking Lot Encoding

[89]:
```
# test_df['type'].unique()
```

[90]:
```
# train_df['type'].unique()
```

[91]:
```python
train_df = pd.get_dummies(
    train_df, columns=["type", "laundry_options", "parking_options"],␣
↪drop_first=True
)

test_df = pd.get_dummies(
    test_df, columns=["type", "laundry_options", "parking_options"],␣
↪drop_first=True
)

# assisted living is not present in test df, ensuring both have same columns
test_df = test_df.reindex(columns=train_df.columns, fill_value=False)

train_df.shape
```

[91]: (332939, 56)

[92]:
```python
# train_df.drop(columns=["type"], inplace=True)
# train_df.drop(columns=["laundry_options"], inplace=True)
# train_df.drop(columns=["parking_options"], inplace=True)

# test_df.drop(columns=["type"], inplace=True)
# test_df.drop(columns=["laundry_options"], inplace=True)
# test_df.drop(columns=["parking_options"], inplace=True)
```

[93]:
```python
train_df.head()
```

[93]:
```
           id                region  price  sqfeet  beds  baths  cats_allowed  \
0  7049965813         orange county   2632    1080     2    2.0             1
1  7036046796        visalia-tulare   1160     768     2    1.0             1
2  7037856890              portland   1262    1075     2    1.0             1
3  7046933042               boulder   1861    1076     2    2.0             1
4  7048650961  sioux falls / SE SD    626     720     1    1.0             1

   dogs_allowed  smoking_allowed  wheelchair_access  …  \
0             1                0                  0  …
1             1                1                  0  …
2             1                0                  0  …
3             1                0                  0  …
4             1                0                  0  …

   laundry_options_laundry on site  laundry_options_no laundry on site  \
0                            False                               False
1                             True                               False
2                             True                               False
3                            False                               False
4                            False                               False
```

```
  laundry_options_w/d hookups  laundry_options_w/d in unit  \
0                       False                         True
1                       False                        False
2                       False                        False
3                       False                         True
4                       False                         True

   parking_options_carport parking_options_detached garage  \
0                    False                           False
1                     True                           False
2                    False                           False
3                    False                            True
4                    False                            True

   parking_options_no parking  parking_options_off-street parking  \
0                       False                               False
1                       False                               False
2                       False                                True
3                       False                               False
4                       False                               False

   parking_options_street parking  parking_options_valet parking
0                           False                          False
1                           False                          False
2                           False                          False
3                           False                          False
4                           False                          False

[5 rows x 56 columns]
```

[94]:
```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 36947 entries, 0 to 38497
Data columns (total 56 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   id                                       36947 non-null  int64
 1   region                                   36947 non-null  object
 2   price                                    36947 non-null  int64
 3   sqfeet                                   36947 non-null  int64
 4   beds                                     36947 non-null  int64
 5   baths                                    36947 non-null  float64
 6   cats_allowed                             36947 non-null  int64
 7   dogs_allowed                             36947 non-null  int64
 8   smoking_allowed                          36947 non-null  int64
 9   wheelchair_access                        36947 non-null  int64
```

```
 10  electric_vehicle_charge            36947 non-null  int64
 11  comes_furnished                    36947 non-null  int64
 12  description                        36947 non-null  object
 13  lat                                36947 non-null  float64
 14  long                               36947 non-null  float64
 15  state                              36947 non-null  object
 16  total_pets                         36947 non-null  int64
 17  luxury_features                    36947 non-null  int64
 18  geo_cluster                        36947 non-null  int32
 19  region_freq                        36947 non-null  float64
 20  log_price                          36947 non-null  float64
 21  log_sqfeet                         36947 non-null  float64
 22  bath_ratio                         36947 non-null  float64
 23  has_outdoor                        36947 non-null  bool
 24  has_pool                           36947 non-null  bool
 25  has_fitness_center                 36947 non-null  bool
 26  mentions_price                     36947 non-null  bool
 27  has_maintenance                    36947 non-null  bool
 28  has_management                     36947 non-null  bool
 29  has_garbage_disposal               36947 non-null  bool
 30  has_closet                         36947 non-null  bool
 31  has_kitchen                        36947 non-null  bool
 32  has_fan                            36947 non-null  bool
 33  has_offer                          36947 non-null  bool
 34  has_appliances                     36947 non-null  bool
 35  type_assisted living               36947 non-null  bool
 36  type_condo                         36947 non-null  bool
 37  type_cottage/cabin                 36947 non-null  bool
 38  type_duplex                        36947 non-null  bool
 39  type_flat                          36947 non-null  bool
 40  type_house                         36947 non-null  bool
 41  type_in-law                        36947 non-null  bool
 42  type_land                          36947 non-null  bool
 43  type_loft                          36947 non-null  bool
 44  type_manufactured                  36947 non-null  bool
 45  type_townhouse                     36947 non-null  bool
 46  laundry_options_laundry on site    36947 non-null  bool
 47  laundry_options_no laundry on site 36947 non-null  bool
 48  laundry_options_w/d hookups        36947 non-null  bool
 49  laundry_options_w/d in unit        36947 non-null  bool
 50  parking_options_carport            36947 non-null  bool
 51  parking_options_detached garage    36947 non-null  bool
 52  parking_options_no parking         36947 non-null  bool
 53  parking_options_off-street parking 36947 non-null  bool
 54  parking_options_street parking     36947 non-null  bool
 55  parking_options_valet parking      36947 non-null  bool
dtypes: bool(33), float64(7), int32(1), int64(12), object(3)
memory usage: 7.8+ MB
```

[95]:
```
train_df.columns
```

[95]:
```
Index(['id', 'region', 'price', 'sqfeet', 'beds', 'baths', 'cats_allowed',
       'dogs_allowed', 'smoking_allowed', 'wheelchair_access',
       'electric_vehicle_charge', 'comes_furnished', 'description', 'lat',
       'long', 'state', 'total_pets', 'luxury_features', 'geo_cluster',
       'region_freq', 'log_price', 'log_sqfeet', 'bath_ratio', 'has_outdoor',
       'has_pool', 'has_fitness_center', 'mentions_price', 'has_maintenance',
       'has_management', 'has_garbage_disposal', 'has_closet', 'has_kitchen',
       'has_fan', 'has_offer', 'has_appliances', 'type_assisted living',
       'type_condo', 'type_cottage/cabin', 'type_duplex', 'type_flat',
       'type_house', 'type_in-law', 'type_land', 'type_loft',
       'type_manufactured', 'type_townhouse',
       'laundry_options_laundry on site', 'laundry_options_no laundry on site',
       'laundry_options_w/d hookups', 'laundry_options_w/d in unit',
       'parking_options_carport', 'parking_options_detached garage',
       'parking_options_no parking', 'parking_options_off-street parking',
       'parking_options_street parking', 'parking_options_valet parking'],
      dtype='object')
```

0.4.9 Region Encoding

[96]:
```
# train_df['region'].value_counts()
```

[97]:
```
#  # Step 1: Compute region-level statistics
# region_stats = train_df.groupby("region").agg({
#     "price": "mean",
#     "sqfeet": "mean",
#     "lat": "mean",
#     "long": "mean"
# }).reset_index()

# # Step 2: Apply K-Means clustering
# num_clusters = 10  # Choose the number of clusters based on analysis
# kmeans = KMeans(n_clusters=num_clusters, random_state=42)
# region_stats["region_cluster"] = kmeans.fit_predict(region_stats[["price",␣
↪"sqfeet", "lat", "long"]])

# # Step 3: Merge the cluster labels back into train_df
# train_df = train_df.merge(region_stats[["region", "region_cluster"]],␣
↪on="region", how="left")

# # Step 4: Drop the original region column
# train_df.drop(columns=["region"], inplace=True)
```

[98]:
```
# # Scatter plot of clusters based on latitude and longitude
# plt.figure(figsize=(12, 8))
```

```
# sns.scatterplot(x=train_df["long"], y=train_df["lat"],␣
↪hue=train_df["region_cluster"], palette="tab10", alpha=0.6, s=50)
# plt.xlabel("Longitude")
# plt.ylabel("Latitude")
# plt.title("Geographical Clustering of Regions")
# plt.legend(title="Region Cluster")
# plt.show()
```

[99]:
```
train_df.shape
```

[99]:
```
(332939, 56)
```

[100]:
```
train_df.head()
```

[100]:
```
           id              region  price  sqfeet  beds  baths  cats_allowed  \
0  7049965813       orange county   2632    1080     2    2.0             1
1  7036046796      visalia-tulare   1160     768     2    1.0             1
2  7037856890            portland   1262    1075     2    1.0             1
3  7046933042             boulder   1861    1076     2    2.0             1
4  7048650961  sioux falls / SE SD    626     720     1    1.0             1

   dogs_allowed  smoking_allowed  wheelchair_access  …  \
0             1                0                  0  …
1             1                1                  0  …
2             1                0                  0  …
3             1                0                  0  …
4             1                0                  0  …

   laundry_options_laundry on site  laundry_options_no laundry on site  \
0                            False                               False
1                             True                               False
2                             True                               False
3                            False                               False
4                            False                               False

  laundry_options_w/d hookups  laundry_options_w/d in unit  \
0                       False                         True
1                       False                        False
2                       False                        False
3                       False                         True
4                       False                         True

   parking_options_carport parking_options_detached garage  \
0                    False                           False
1                     True                           False
2                    False                           False
3                    False                            True
```

```
4                    False                              True

   parking_options_no parking  parking_options_off-street parking  \
0                       False                               False
1                       False                               False
2                       False                                True
3                       False                               False
4                       False                               False

   parking_options_street parking  parking_options_valet parking
0                           False                          False
1                           False                          False
2                           False                          False
3                           False                          False
4                           False                          False

[5 rows x 56 columns]
```

0.4.10 State Encoding

0.4.11 5. Normalize and Scale Numerical Data

0.4.12 5.1. Scale continous variables

[101]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 332939 entries, 0 to 346478
Data columns (total 56 columns):
 #   Column                                   Non-Null Count   Dtype
---  ------                                   --------------   -----
 0   id                                       332939 non-null  int64
 1   region                                   332939 non-null  object
 2   price                                    332939 non-null  int64
 3   sqfeet                                   332939 non-null  int64
 4   beds                                     332939 non-null  int64
 5   baths                                    332939 non-null  float64
 6   cats_allowed                             332939 non-null  int64
 7   dogs_allowed                             332939 non-null  int64
 8   smoking_allowed                          332939 non-null  int64
 9   wheelchair_access                        332939 non-null  int64
 10  electric_vehicle_charge                  332939 non-null  int64
 11  comes_furnished                          332939 non-null  int64
 12  description                              332939 non-null  object
 13  lat                                      332939 non-null  float64
 14  long                                     332939 non-null  float64
 15  state                                    332939 non-null  object
 16  total_pets                               332939 non-null  int64
 17  luxury_features                          332939 non-null  int64
```

```
 18  geo_cluster                        332939 non-null  int32
 19  region_freq                        332939 non-null  float64
 20  log_price                          332939 non-null  float64
 21  log_sqfeet                         332939 non-null  float64
 22  bath_ratio                         332939 non-null  float64
 23  has_outdoor                        332939 non-null  bool
 24  has_pool                           332939 non-null  bool
 25  has_fitness_center                 332939 non-null  bool
 26  mentions_price                     332939 non-null  bool
 27  has_maintenance                    332939 non-null  bool
 28  has_management                     332939 non-null  bool
 29  has_garbage_disposal               332939 non-null  bool
 30  has_closet                         332939 non-null  bool
 31  has_kitchen                        332939 non-null  bool
 32  has_fan                            332939 non-null  bool
 33  has_offer                          332939 non-null  bool
 34  has_appliances                     332939 non-null  bool
 35  type_assisted living               332939 non-null  bool
 36  type_condo                         332939 non-null  bool
 37  type_cottage/cabin                 332939 non-null  bool
 38  type_duplex                        332939 non-null  bool
 39  type_flat                          332939 non-null  bool
 40  type_house                         332939 non-null  bool
 41  type_in-law                        332939 non-null  bool
 42  type_land                          332939 non-null  bool
 43  type_loft                          332939 non-null  bool
 44  type_manufactured                  332939 non-null  bool
 45  type_townhouse                     332939 non-null  bool
 46  laundry_options_laundry on site    332939 non-null  bool
 47  laundry_options_no laundry on site  332939 non-null  bool
 48  laundry_options_w/d hookups        332939 non-null  bool
 49  laundry_options_w/d in unit        332939 non-null  bool
 50  parking_options_carport            332939 non-null  bool
 51  parking_options_detached garage    332939 non-null  bool
 52  parking_options_no parking         332939 non-null  bool
 53  parking_options_off-street parking  332939 non-null  bool
 54  parking_options_street parking     332939 non-null  bool
 55  parking_options_valet parking      332939 non-null  bool
dtypes: bool(33), float64(7), int32(1), int64(12), object(3)
memory usage: 70.2+ MB
```

[102]:

```python
scaler = StandardScaler()

# selecting numerical columns except the price
num_cols = train_df.select_dtypes(include=['int32', 'int64', 'float64']).columns
num_cols = num_cols.drop('price')
num_cols = num_cols.drop('log_price')
```

```
# keeping unscaled for Decision tree models
train_df_unscaled = train_df.copy()
test_df_unscaled = test_df.copy()

train_df[num_cols] = scaler.fit_transform(train_df[num_cols])
test_df[num_cols] = scaler.fit_transform(test_df[num_cols])
```

[103]:
```
train_df[num_cols].head()
```

[103]:
```
         id    sqfeet      beds    baths  cats_allowed  dogs_allowed  \
0  1.023725  0.197807  0.063173  0.864745      0.611957      0.640172
1 -0.556416 -0.659348  0.063173 -0.839095      0.611957      0.640172
2 -0.350927  0.184071  0.063173 -0.839095      0.611957      0.640172
3  0.679433  0.186818  0.063173  0.864745      0.611957      0.640172
4  0.874458 -0.791218 -1.151463 -0.839095      0.611957      0.640172

   smoking_allowed  wheelchair_access  electric_vehicle_charge  \
0        -1.669306          -0.293298                -0.112677
1         0.599051          -0.293298                -0.112677
2        -1.669306          -0.293298                -0.112677
3        -1.669306          -0.293298                 8.874963
4        -1.669306          -0.293298                -0.112677

   comes_furnished       lat      long  total_pets  luxury_features  \
0         -0.21444 -0.613742 -1.541101    0.644381        -0.336857
1         -0.21444 -0.165086 -1.631841    0.644381        -0.336857
2         -0.21444  1.498379 -1.822695    0.644381        -0.336857
3         -0.21444  0.497235 -0.764054    0.644381         2.145337
4         -0.21444  1.136100 -0.259375    0.644381        -0.336857

   geo_cluster  region_freq  log_sqfeet  bath_ratio
0    -0.694380     0.538839    0.381266    0.713428
1    -0.694380    -1.606406   -0.669772   -1.293978
2    -0.694380     0.661273    0.366957   -1.293978
3     2.255255     0.887508    0.369824    0.713428
4    -0.325675    -0.728080   -0.868689    0.713428
```

[104]:
```
train_df.drop(columns=["region"], inplace=True)
train_df.drop(columns=["description"], inplace=True)
train_df.drop(columns=["state"], inplace=True)
train_df.drop(columns=["sqfeet"], inplace=True)

test_df.drop(columns=["region"], inplace=True)
test_df.drop(columns=["description"], inplace=True)
test_df.drop(columns=["state"], inplace=True)
test_df.drop(columns=["sqfeet"], inplace=True)
```

```python
# unscaled

train_df_unscaled.drop(columns=["region"], inplace=True)
train_df_unscaled.drop(columns=["description"], inplace=True)
train_df_unscaled.drop(columns=["state"], inplace=True)
train_df_unscaled.drop(columns=["sqfeet"], inplace=True)

test_df_unscaled.drop(columns=["region"], inplace=True)
test_df_unscaled.drop(columns=["description"], inplace=True)
test_df_unscaled.drop(columns=["state"], inplace=True)
test_df_unscaled.drop(columns=["sqfeet"], inplace=True)
```

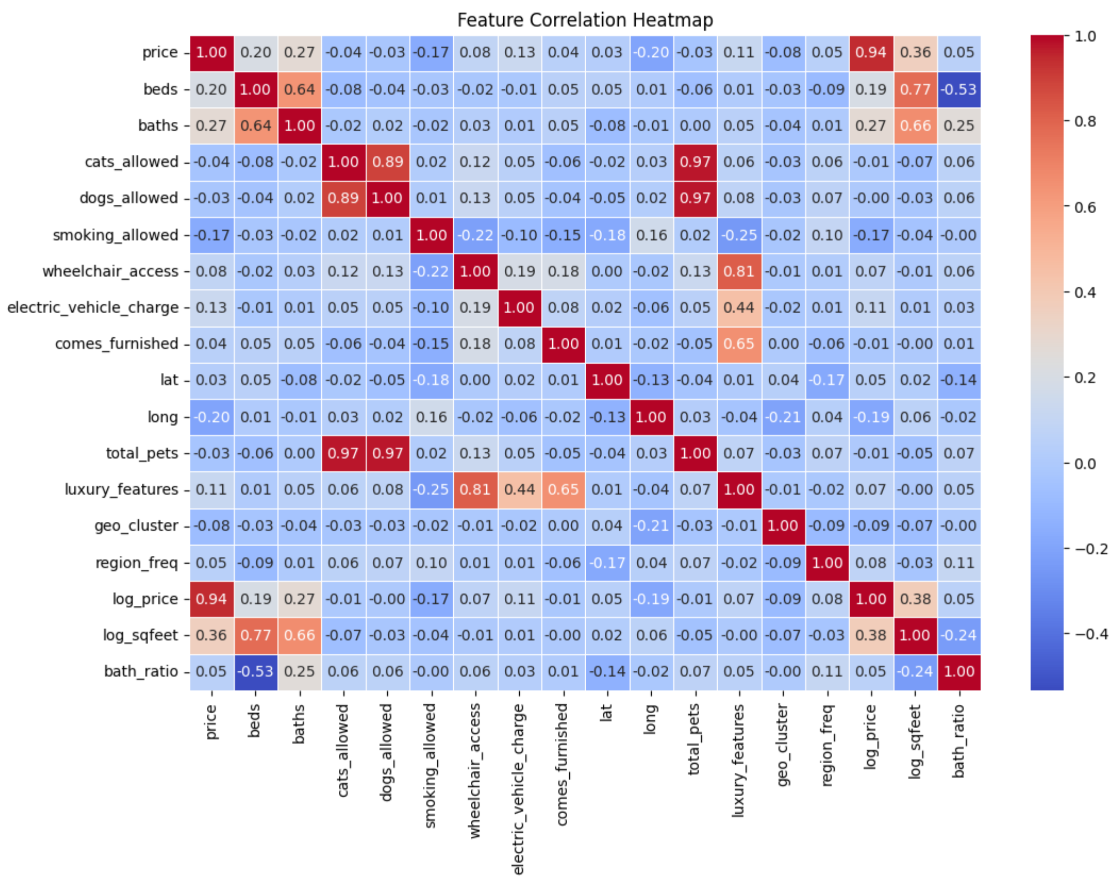
[105]:
```python
train_df.info()
```

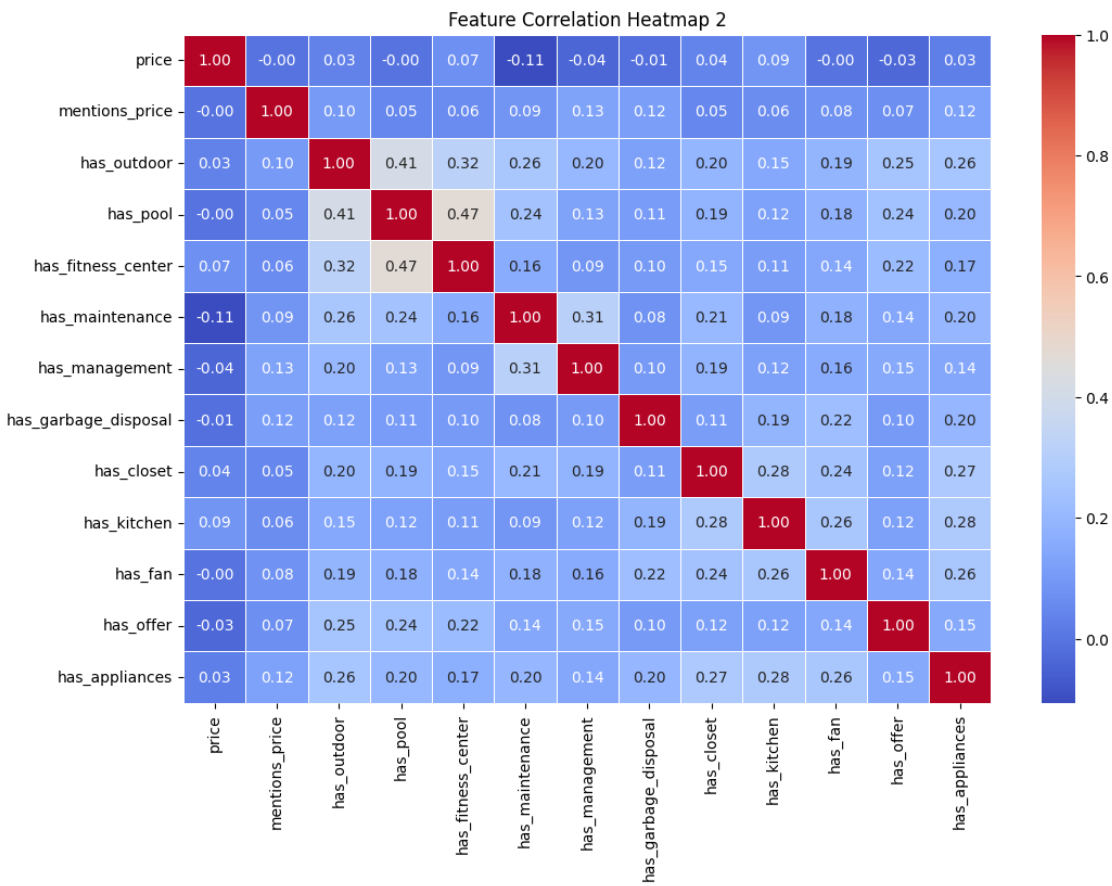
```
<class 'pandas.core.frame.DataFrame'>
Index: 332939 entries, 0 to 346478
Data columns (total 52 columns):
 #   Column                                   Non-Null Count   Dtype
---  ------                                   --------------   -----
 0   id                                       332939 non-null  float64
 1   price                                    332939 non-null  int64
 2   beds                                     332939 non-null  float64
 3   baths                                    332939 non-null  float64
 4   cats_allowed                             332939 non-null  float64
 5   dogs_allowed                             332939 non-null  float64
 6   smoking_allowed                          332939 non-null  float64
 7   wheelchair_access                        332939 non-null  float64
 8   electric_vehicle_charge                  332939 non-null  float64
 9   comes_furnished                          332939 non-null  float64
 10  lat                                      332939 non-null  float64
 11  long                                     332939 non-null  float64
 12  total_pets                               332939 non-null  float64
 13  luxury_features                          332939 non-null  float64
 14  geo_cluster                              332939 non-null  float64
 15  region_freq                              332939 non-null  float64
 16  log_price                                332939 non-null  float64
 17  log_sqfeet                               332939 non-null  float64
 18  bath_ratio                               332939 non-null  float64
 19  has_outdoor                              332939 non-null  bool
 20  has_pool                                 332939 non-null  bool
 21  has_fitness_center                       332939 non-null  bool
 22  mentions_price                           332939 non-null  bool
 23  has_maintenance                          332939 non-null  bool
 24  has_management                           332939 non-null  bool
 25  has_garbage_disposal                     332939 non-null  bool
 26  has_closet                               332939 non-null  bool
 27  has_kitchen                              332939 non-null  bool
```

```
 28  has_fan                          332939 non-null  bool
 29  has_offer                        332939 non-null  bool
 30  has_appliances                   332939 non-null  bool
 31  type_assisted living             332939 non-null  bool
 32  type_condo                       332939 non-null  bool
 33  type_cottage/cabin               332939 non-null  bool
 34  type_duplex                      332939 non-null  bool
 35  type_flat                        332939 non-null  bool
 36  type_house                       332939 non-null  bool
 37  type_in-law                      332939 non-null  bool
 38  type_land                        332939 non-null  bool
 39  type_loft                        332939 non-null  bool
 40  type_manufactured                332939 non-null  bool
 41  type_townhouse                   332939 non-null  bool
 42  laundry_options_laundry on site  332939 non-null  bool
 43  laundry_options_no laundry on site 332939 non-null  bool
 44  laundry_options_w/d hookups      332939 non-null  bool
 45  laundry_options_w/d in unit      332939 non-null  bool
 46  parking_options_carport          332939 non-null  bool
 47  parking_options_detached garage  332939 non-null  bool
 48  parking_options_no parking       332939 non-null  bool
 49  parking_options_off-street parking 332939 non-null  bool
 50  parking_options_street parking   332939 non-null  bool
 51  parking_options_valet parking    332939 non-null  bool
dtypes: bool(33), float64(18), int64(1)
memory usage: 61.3 MB
```

[106]:
```
# selected_cols = ["price", "sqfeet", "beds", "baths", "lat", "long",␣
↪"price_per_sqft", "total_pets", "luxury_features", "region_cluster"]
selected_columns = [
    "price", "beds", "baths", "cats_allowed", "dogs_allowed",
    "smoking_allowed", "wheelchair_access", "electric_vehicle_charge",␣
↪"comes_furnished",
    "lat", "long", "total_pets", "luxury_features", "geo_cluster",
    "region_freq", "log_price", "log_sqfeet", "bath_ratio"
]

selected_columns2 = [
    "price", "mentions_price", "has_outdoor", "has_pool", "has_fitness_center",
    "has_maintenance", "has_management", "has_garbage_disposal", "has_closet",␣
↪"has_kitchen", "has_fan", "has_offer", "has_appliances"
]


# Compute the correlation matrix
corr_matrix = train_df[selected_columns].corr()
corr_matrix2 = train_df[selected_columns2].corr()
```

```python
# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()

plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix2, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.
↪5)
plt.title("Feature Correlation Heatmap 2")
plt.show()
```





Key Takeaway - Cats and Dogs can be dropped, high correlation with total pets - `bath ratio`, `Wheelchair access`, `Comes furnished` can be dropped, low impact on price

[107]:
```
train_df.drop(columns=["cats_allowed"], inplace=True)
test_df.drop(columns=["cats_allowed"], inplace=True)
train_df_unscaled.drop(columns=["cats_allowed"], inplace=True)
test_df_unscaled.drop(columns=["cats_allowed"], inplace=True)
```

[108]:
```
train_df.drop(columns=["dogs_allowed"], inplace=True)
test_df.drop(columns=["dogs_allowed"], inplace=True)
train_df_unscaled.drop(columns=["dogs_allowed"], inplace=True)
test_df_unscaled.drop(columns=["dogs_allowed"], inplace=True)
```

[109]:
```
train_df.drop(columns=["bath_ratio"], inplace=True)
test_df.drop(columns=["bath_ratio"], inplace=True)
train_df_unscaled.drop(columns=["bath_ratio"], inplace=True)
test_df_unscaled.drop(columns=["bath_ratio"], inplace=True)
```

[110]:
```
train_df.drop(columns=["wheelchair_access"], inplace=True)
test_df.drop(columns=["wheelchair_access"], inplace=True)
train_df_unscaled.drop(columns=["wheelchair_access"], inplace=True)
test_df_unscaled.drop(columns=["wheelchair_access"], inplace=True)
```

[111]:
```
train_df.drop(columns=["comes_furnished"], inplace=True)
test_df.drop(columns=["comes_furnished"], inplace=True)
train_df_unscaled.drop(columns=["comes_furnished"], inplace=True)
test_df_unscaled.drop(columns=["comes_furnished"], inplace=True)
```

[112]:
```
train_df.drop(columns=["mentions_price"], inplace=True)
test_df.drop(columns=["mentions_price"], inplace=True)
train_df_unscaled.drop(columns=["mentions_price"], inplace=True)
test_df_unscaled.drop(columns=["mentions_price"], inplace=True)
```

[113]:
```
train_df.drop(columns=["has_pool"], inplace=True)
test_df.drop(columns=["has_pool"], inplace=True)
train_df_unscaled.drop(columns=["has_pool"], inplace=True)
test_df_unscaled.drop(columns=["has_pool"], inplace=True)
```

[114]:
```
train_df.drop(columns=["has_garbage_disposal"], inplace=True)
test_df.drop(columns=["has_garbage_disposal"], inplace=True)
train_df_unscaled.drop(columns=["has_garbage_disposal"], inplace=True)
test_df_unscaled.drop(columns=["has_garbage_disposal"], inplace=True)
```

[115]:
```
train_df.drop(columns=["has_fan"], inplace=True)
test_df.drop(columns=["has_fan"], inplace=True)
train_df_unscaled.drop(columns=["has_fan"], inplace=True)
test_df_unscaled.drop(columns=["has_fan"], inplace=True)
```

[116]:
```
train_df.head()
```

[116]:
```
         id  price      beds     baths  smoking_allowed  \
0  1.023725   2632  0.063173  0.864745        -1.669306
1 -0.556416   1160  0.063173 -0.839095         0.599051
2 -0.350927   1262  0.063173 -0.839095        -1.669306
3  0.679433   1861  0.063173  0.864745        -1.669306
4  0.874458    626 -1.151463 -0.839095        -1.669306

   electric_vehicle_charge       lat      long  total_pets  luxury_features  \
0                -0.112677 -0.613742 -1.541101    0.644381        -0.336857
1                -0.112677 -0.165086 -1.631841    0.644381        -0.336857
2                -0.112677  1.498379 -1.822695    0.644381        -0.336857
3                 8.874963  0.497235 -0.764054    0.644381         2.145337
4                -0.112677  1.136100 -0.259375    0.644381        -0.336857

   …  laundry_options_laundry on site  laundry_options_no laundry on site  \
```

```
0  …                           False                             False
1  …                            True                             False
2  …                            True                             False
3  …                           False                             False
4  …                           False                             False

   laundry_options_w/d hookups  laundry_options_w/d in unit  \
0                        False                         True
1                        False                        False
2                        False                        False
3                        False                         True
4                        False                         True

   parking_options_carport  parking_options_detached garage  \
0                    False                            False
1                     True                            False
2                    False                            False
3                    False                             True
4                    False                             True

   parking_options_no parking  parking_options_off-street parking  \
0                       False                               False
1                       False                               False
2                       False                                True
3                       False                               False
4                       False                               False

   parking_options_street parking  parking_options_valet parking
0                           False                          False
1                           False                          False
2                           False                          False
3                           False                          False
4                           False                          False

[5 rows x 43 columns]
```

[117]:
```
train_df_unscaled.head()
```

[117]:
```
           id  price  beds  baths  smoking_allowed  electric_vehicle_charge  \
0  7049965813   2632     2    2.0                0                        0
1  7036046796   1160     2    1.0                1                        0
2  7037856890   1262     2    1.0                0                        0
3  7046933042   1861     2    2.0                0                        1
4  7048650961    626     1    1.0                0                        0

       lat      long  total_pets  luxury_features  …  \
0  33.8123 -117.8530           2                0  …
```

```
1  36.3008 -119.3440           2                0  …
2  45.5273 -122.4800           2                0  …
3  39.9744 -105.0850           2                1  …
4  43.5179  -96.7924           2                0  …

   laundry_options_laundry on site  laundry_options_no laundry on site  \
0                            False                               False
1                             True                               False
2                             True                               False
3                            False                               False
4                            False                               False

   laundry_options_w/d hookups  laundry_options_w/d in unit  \
0                        False                         True
1                        False                        False
2                        False                        False
3                        False                         True
4                        False                         True

   parking_options_carport  parking_options_detached garage  \
0                    False                            False
1                     True                            False
2                    False                            False
3                    False                             True
4                    False                             True

   parking_options_no parking  parking_options_off-street parking  \
0                       False                               False
1                       False                               False
2                       False                                True
3                       False                               False
4                       False                               False

   parking_options_street parking  parking_options_valet parking
0                           False                          False
1                           False                          False
2                           False                          False
3                           False                          False
4                           False                          False

[5 rows x 43 columns]
```

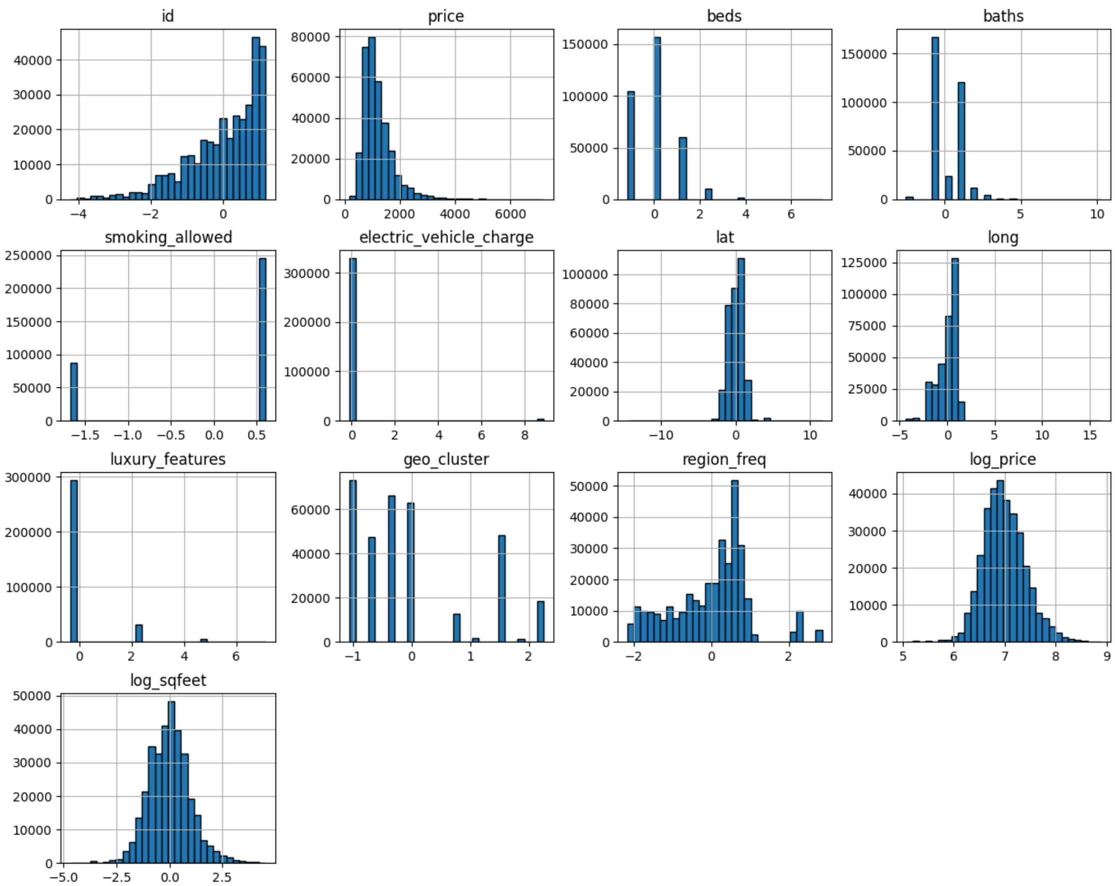
[118]:

```
# region_freq: 0.05
# geo_cluster: 0.08
# total_pets: -0.03
# lat: 0.03
# has_offer: -0.03
```

```python
# has_appliances: 0.03
# has_outdoor: 0.03

train_df.drop(columns=["total_pets"], inplace=True)
test_df.drop(columns=["total_pets"], inplace=True)
train_df_unscaled.drop(columns=["total_pets"], inplace=True)
test_df_unscaled.drop(columns=["total_pets"], inplace=True)

train_df.drop(columns=["has_offer"], inplace=True)
test_df.drop(columns=["has_offer"], inplace=True)
train_df_unscaled.drop(columns=["has_offer"], inplace=True)
test_df_unscaled.drop(columns=["has_offer"], inplace=True)

train_df.drop(columns=["has_appliances"], inplace=True)
test_df.drop(columns=["has_appliances"], inplace=True)
train_df_unscaled.drop(columns=["has_appliances"], inplace=True)
test_df_unscaled.drop(columns=["has_appliances"], inplace=True)

train_df.drop(columns=["has_outdoor"], inplace=True)
test_df.drop(columns=["has_outdoor"], inplace=True)
train_df_unscaled.drop(columns=["has_outdoor"], inplace=True)
test_df_unscaled.drop(columns=["has_outdoor"], inplace=True)
```

[119]:
```python
train_df.hist(figsize=(15, 12), bins=30, edgecolor="black")
```

[119]:
```
array([[<Axes: title={'center': 'id'}>,
        <Axes: title={'center': 'price'}>,
        <Axes: title={'center': 'beds'}>,
        <Axes: title={'center': 'baths'}>],
       [<Axes: title={'center': 'smoking_allowed'}>,
        <Axes: title={'center': 'electric_vehicle_charge'}>,
        <Axes: title={'center': 'lat'}>,
        <Axes: title={'center': 'long'}>],
       [<Axes: title={'center': 'luxury_features'}>,
        <Axes: title={'center': 'geo_cluster'}>,
        <Axes: title={'center': 'region_freq'}>,
        <Axes: title={'center': 'log_price'}>],
       [<Axes: title={'center': 'log_sqfeet'}>, <Axes: >, <Axes: >,
        <Axes: >]], dtype=object)
```



0.5 Building Models

[120]:
```
train_df.shape
```

[120]: (332939, 39)

[121]:
```
test_df.shape
```

[121]: (36947, 39)

[122]:
```
# PBP(train_df_unscaled)
```

[123]:
```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 332939 entries, 0 to 346478
Data columns (total 39 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   id                              332939 non-null  float64
```

```
 1   price                              332939 non-null  int64
 2   beds                               332939 non-null  float64
 3   baths                              332939 non-null  float64
 4   smoking_allowed                    332939 non-null  float64
 5   electric_vehicle_charge            332939 non-null  float64
 6   lat                                332939 non-null  float64
 7   long                               332939 non-null  float64
 8   luxury_features                    332939 non-null  float64
 9   geo_cluster                        332939 non-null  float64
 10  region_freq                        332939 non-null  float64
 11  log_price                          332939 non-null  float64
 12  log_sqfeet                         332939 non-null  float64
 13  has_fitness_center                 332939 non-null  bool
 14  has_maintenance                    332939 non-null  bool
 15  has_management                     332939 non-null  bool
 16  has_closet                         332939 non-null  bool
 17  has_kitchen                        332939 non-null  bool
 18  type_assisted living               332939 non-null  bool
 19  type_condo                         332939 non-null  bool
 20  type_cottage/cabin                 332939 non-null  bool
 21  type_duplex                        332939 non-null  bool
 22  type_flat                          332939 non-null  bool
 23  type_house                         332939 non-null  bool
 24  type_in-law                        332939 non-null  bool
 25  type_land                          332939 non-null  bool
 26  type_loft                          332939 non-null  bool
 27  type_manufactured                  332939 non-null  bool
 28  type_townhouse                     332939 non-null  bool
 29  laundry_options_laundry on site    332939 non-null  bool
 30  laundry_options_no laundry on site 332939 non-null  bool
 31  laundry_options_w/d hookups        332939 non-null  bool
 32  laundry_options_w/d in unit        332939 non-null  bool
 33  parking_options_carport            332939 non-null  bool
 34  parking_options_detached garage    332939 non-null  bool
 35  parking_options_no parking         332939 non-null  bool
 36  parking_options_off-street parking 332939 non-null  bool
 37  parking_options_street parking     332939 non-null  bool
 38  parking_options_valet parking      332939 non-null  bool
dtypes: bool(26), float64(12), int64(1)
memory usage: 43.8 MB
```

0.5.1 1. Linear Regression with Log Price

[124]:
```python
X_train = train_df.drop(columns=["log_price", "price"], axis=1)
y_train = train_df['log_price']

X_test = test_df.drop(columns=["log_price", "price"], axis=1)
```

```
y_test = test_df['log_price']
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[125]:
```
# X_train.head()
```

[122]:
```
# X_train.info()
```

[123]:
```
# X_test.info()
```

[124]:
```
# y_train.head()
```

[125]:
```
# y_test.head()
```

[126]:
```
# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

[126]:
```
LinearRegression()
```

[127]:
```
# Make predictions
y_pred_log = model.predict(X_test)
y_pred_price = np.expm1(y_pred_log)
```

[128]:
```
mae = mean_absolute_error(np.expm1(y_test), np.expm1(y_pred_log))  # Convert␣
↪log_price back to price
mse = mean_squared_error(y_test, y_pred_log)
r2 = r2_score(y_test, y_pred_log)

print(f"Model Performance:")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"R² Score: {r2:.4f}")
```

```
Model Performance:
Mean Absolute Error (MAE): 314.7265
Mean Squared Error (MSE): 0.1171
R² Score: 0.3149
```

[129]:
```
predictions_df = pd.DataFrame({"Actual Price": np.expm1(y_test), "Predicted␣
↪Price": y_pred_price})
predictions_df.head(10)
```

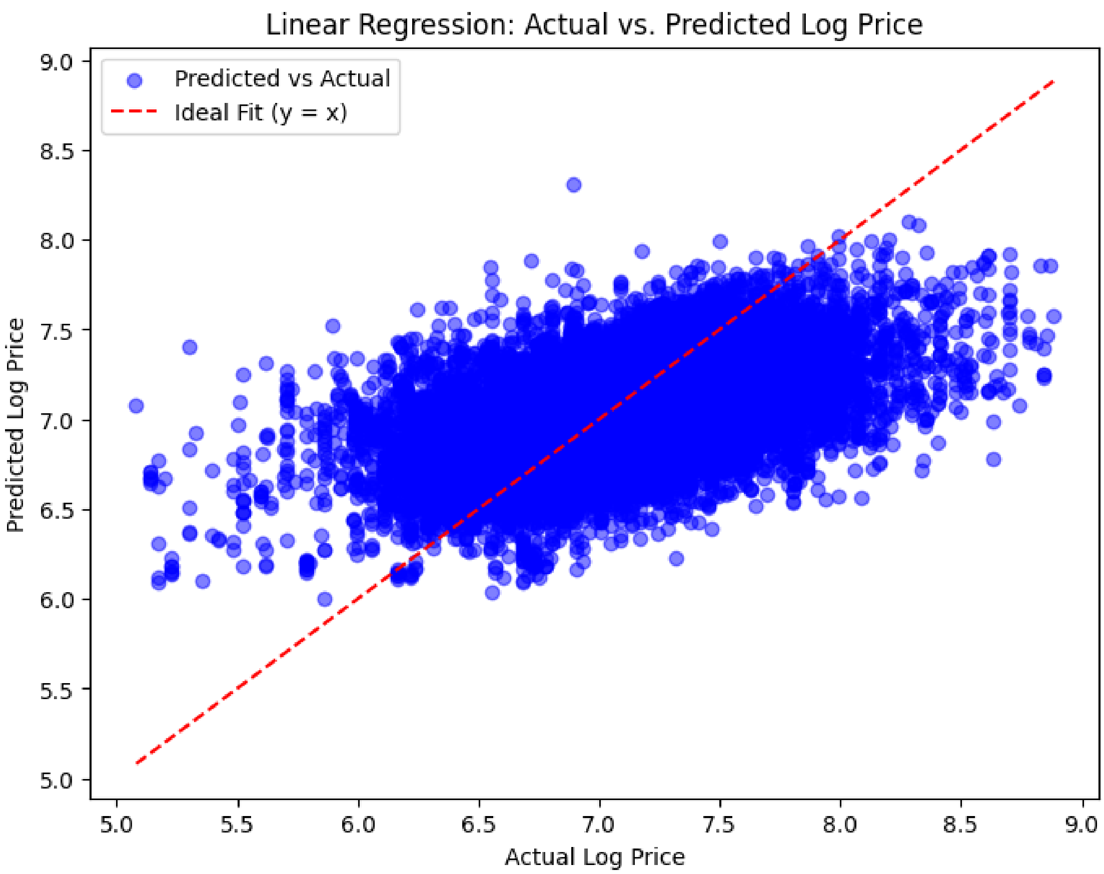
[129]:

	Actual Price	Predicted Price
0	1750.0	959.398623
1	850.0	1347.971394
2	1500.0	1438.245722
3	899.0	957.427268

```
5          899.0     1232.364281
6          805.0     1027.897200
7         1300.0      949.931136
8          735.0      453.393813
9         1070.0      927.160077
10         805.0      863.142969
```

[130]:

```python
# Scatter plot of actual vs. predicted log_price values
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred_log, alpha=0.5, label="Predicted vs Actual",␣
 ↪color="blue")

# Plot ideal prediction line (y = x) for reference
min_val = min(min(y_test), min(y_pred_log))
max_val = max(max(y_test), max(y_pred_log))
plt.plot([min_val, max_val], [min_val, max_val], linestyle="--", color="red",␣
 ↪label="Ideal Fit (y = x)")

# Labels and title
plt.xlabel("Actual Log Price")
plt.ylabel("Predicted Log Price")
plt.title("Linear Regression: Actual vs. Predicted Log Price")
plt.legend()
plt.show()
```



0.5.2 1.2 Linear Regression with Price

[131]:
```
X_train = train_df.drop(columns=["log_price", "price"], axis=1)
y_train = train_df['price']

X_test = test_df.drop(columns=["log_price", "price"], axis=1)
y_test = test_df['price']
```

[132]:
```
# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

[132]: LinearRegression()

[133]:
```
# Make predictions on the test set
y_pred = model.predict(X_test)
```

[134]:
```
# Calculate evaluation metrics
r2 = r2_score(y_test, y_pred)
```

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

# Print results
print(f"Model Evaluation Metrics for Price Prediction:")
print(f"  R² Score: {r2:.4f}  (Higher is better, max = 1)")
print(f"  MAE: ${mae:.2f} (Lower is better, avg absolute error)")
print(f"  MSE: {mse:.2f} (Lower is better, penalizes large errors)")
print(f"  RMSE: ${rmse:.2f} (Lower is better, interpretable error in price␣
↪units)")
```

```
Model Evaluation Metrics for Price Prediction:
  R² Score: 0.2858  (Higher is better, max = 1)
  MAE: $327.90 (Lower is better, avg absolute error)
  MSE: 235255.06 (Lower is better, penalizes large errors)
  RMSE: $485.03 (Lower is better, interpretable error in price units)
```

[135]:

```
predictions_df = pd.DataFrame({"Actual Price": y_test, "Predicted Price":␣
↪y_pred})
predictions_df.head(10)
```

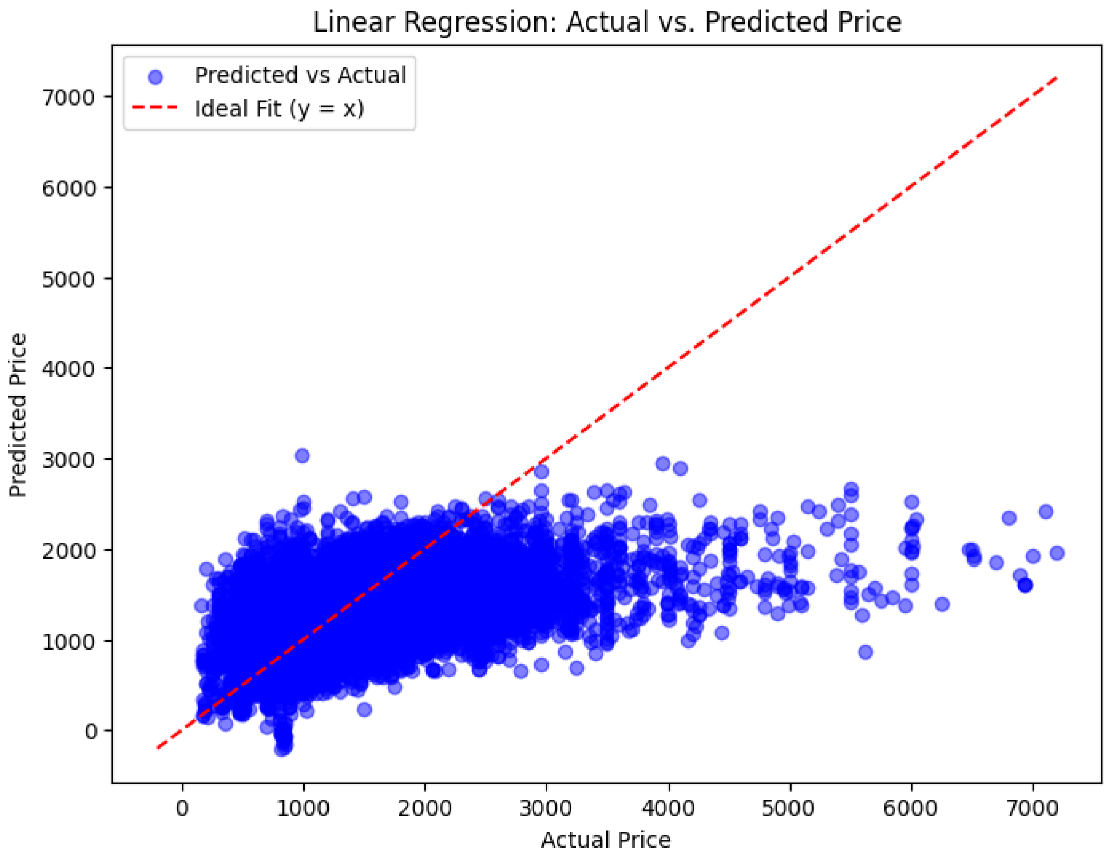
[135]:

	Actual Price	Predicted Price
0	1750	976.421006
1	850	1540.127956
2	1500	1485.787601
3	899	1036.872606
5	899	1236.041307
6	805	1104.277259
7	1300	1028.157372
8	735	120.250520
9	1070	870.715609
10	805	852.420340

[136]:

```
# Scatter plot of actual vs. predicted price values
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5, label="Predicted vs Actual",␣
↪color="blue")

# Plot ideal prediction line (y = x) for reference
min_val = min(min(y_test), min(y_pred))
max_val = max(max(y_test), max(y_pred))
plt.plot([min_val, max_val], [min_val, max_val], linestyle="--", color="red",␣
↪label="Ideal Fit (y = x)")

# Labels and title
plt.xlabel("Actual Price")
```

```
plt.ylabel("Predicted Price")
plt.title("Linear Regression: Actual vs. Predicted Price")
plt.legend()
plt.show()
```



0.5.3 2. RandomForest with Price

[141]:
```
X_train = train_df_unscaled.drop(columns=["log_price", "price"], axis=1)
y_train = train_df_unscaled['price']

X_test = test_df_unscaled.drop(columns=["log_price", "price"], axis=1)
y_test = test_df_unscaled['price']
```

[142]:
```
# Initialize and train the Random Forest Regressor model
randomforest_model = RandomForestRegressor(n_estimators=100, random_state=42,␣
↪n_jobs=-1)
randomforest_model.fit(X_train, y_train)
```

[142]: RandomForestRegressor(n_jobs=-1, random_state=42)

[143]:
```
# Make predictions on the test set
randomforest_y_pred = randomforest_model.predict(X_test)
```

[144]:
```
# Calculate evaluation metrics
r2 = r2_score(y_test, randomforest_y_pred)
mae = mean_absolute_error(y_test, randomforest_y_pred)
mse = mean_squared_error(y_test, randomforest_y_pred)
rmse = np.sqrt(mse)

# Print results
print(f"Random Forest Model Evaluation Metrics for Price Prediction:")
print(f"  R² Score: {r2:.4f}  (Higher is better, max = 1)")
print(f"  MAE: ${mae:.2f} (Lower is better, avg absolute error)")
print(f"  MSE: {mse:.2f} (Lower is better, penalizes large errors)")
print(f"  RMSE: ${rmse:.2f} (Lower is better, interpretable error in price␣
↪units)")
```

```
Random Forest Model Evaluation Metrics for Price Prediction:
  R² Score: 0.8251  (Higher is better, max = 1)
  MAE: $118.99 (Lower is better, avg absolute error)
  MSE: 57621.63 (Lower is better, penalizes large errors)
  RMSE: $240.05 (Lower is better, interpretable error in price units)
```

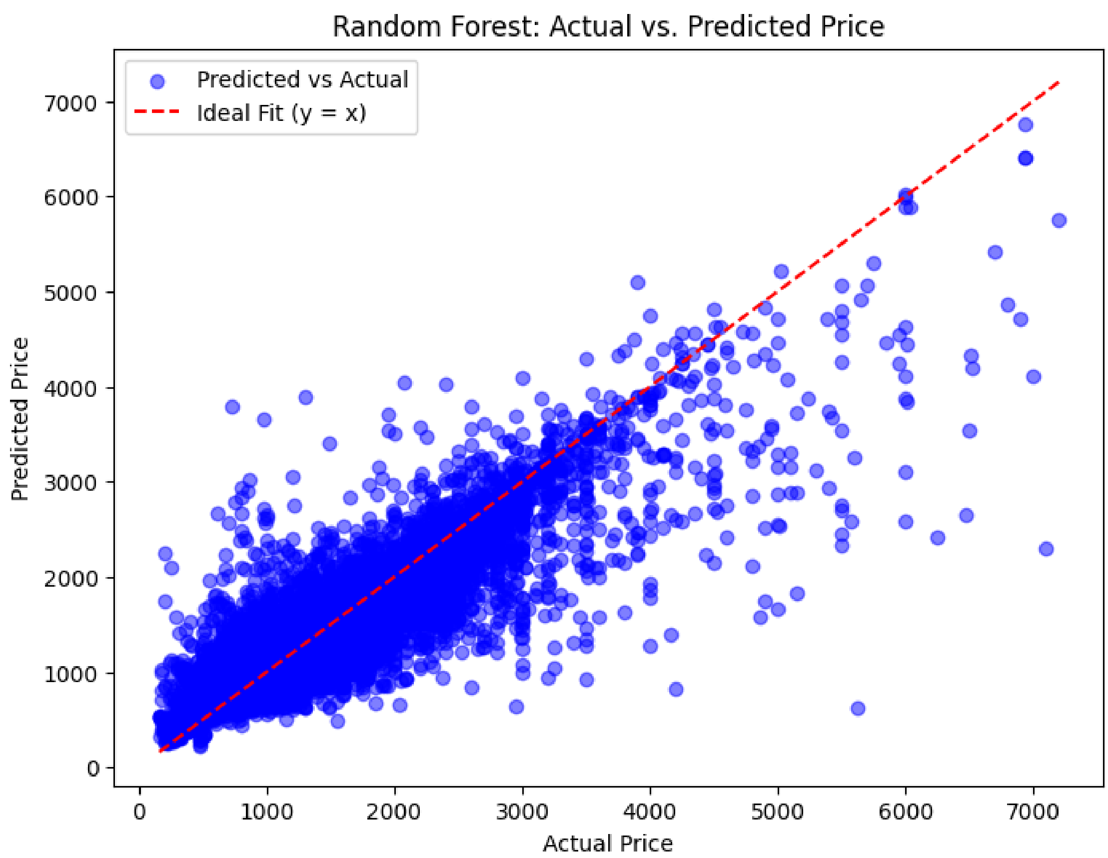
[145]:
```
predictions_df = pd.DataFrame({"Actual Price": y_test, "Predicted Price":␣
↪randomforest_y_pred})
predictions_df.head(10)
```

[145]:

	Actual Price	Predicted Price
0	1750	1596.45
1	850	917.54
2	1500	1500.70
3	899	718.17
5	899	912.95
6	805	967.31
7	1300	1295.96
8	735	653.30
9	1070	1068.20
10	805	633.85

[146]:
```
# Scatter plot of actual vs. predicted price values
plt.figure(figsize=(8, 6))
plt.scatter(y_test, randomforest_y_pred, alpha=0.5, label="Predicted vs␣
↪Actual", color="blue")

# Plot ideal prediction line (y = x) for reference
min_val = min(min(y_test), min(randomforest_y_pred))
max_val = max(max(y_test), max(randomforest_y_pred))
```

```
plt.plot([min_val, max_val], [min_val, max_val], linestyle="--", color="red",␣
 ↪label="Ideal Fit (y = x)")

# Labels and title
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Random Forest: Actual vs. Predicted Price")
plt.legend()
plt.show()
```



0.5.4 3. XGBoost

[130]:
```
X_train = train_df_unscaled.drop(columns=["log_price", "price"], axis=1)
y_train = train_df_unscaled['price']

X_test = test_df_unscaled.drop(columns=["log_price", "price"], axis=1)
y_test = test_df_unscaled['price']
```

[131]:
```python
xgb_model = XGBRegressor(n_estimators=360, learning_rate=0.1, random_state=42)
xgb_model.fit(X_train, y_train)
```

[131]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=360, n_jobs=None,
             num_parallel_tree=None, random_state=42, …)
```

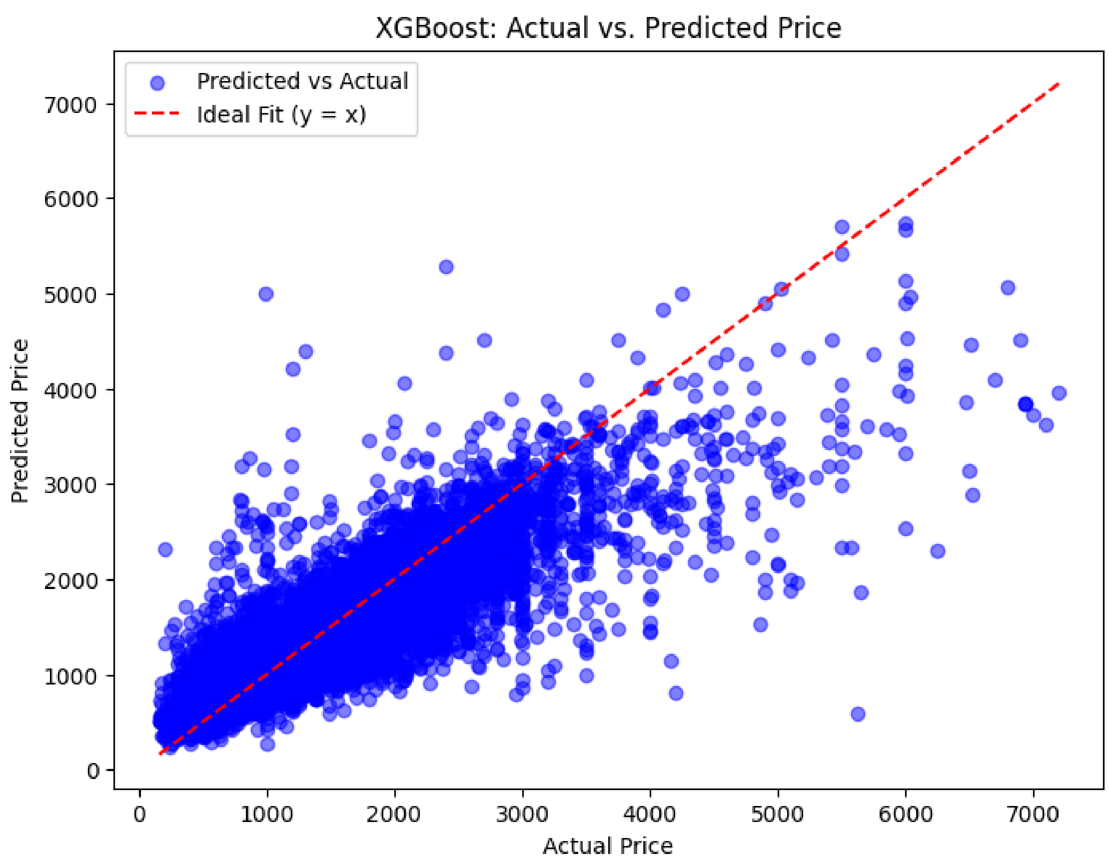
[132]:
```python
# Make predictions on the test set
xgb_preds = xgb_model.predict(X_test)
```

[133]:
```python
# Calculate evaluation metrics
r2 = r2_score(y_test, xgb_preds)
mae = mean_absolute_error(y_test, xgb_preds)
mse = mean_squared_error(y_test, xgb_preds)
rmse = np.sqrt(mse)

# Print results
print(f"Random Forest Model Evaluation Metrics for Price Prediction:")
print(f"  R² Score: {r2:.4f}  (Higher is better, max = 1)")
print(f"  MAE: ${mae:.2f} (Lower is better, avg absolute error)")
print(f"  MSE: {mse:.2f} (Lower is better, penalizes large errors)")
print(f"  RMSE: ${rmse:.2f} (Lower is better, interpretable error in price␣
 ↪units)")
```

```
Random Forest Model Evaluation Metrics for Price Prediction:
  R² Score: 0.7555  (Higher is better, max = 1)
  MAE: $173.24 (Lower is better, avg absolute error)
  MSE: 80552.12 (Lower is better, penalizes large errors)
  RMSE: $283.82 (Lower is better, interpretable error in price units)
```

[134]:
```python
# Scatter plot of actual vs. predicted price values
plt.figure(figsize=(8, 6))
plt.scatter(y_test, xgb_preds, alpha=0.5, label="Predicted vs Actual",␣
 ↪color="blue")

# Plot ideal prediction line (y = x) for reference
min_val = min(min(y_test), min(xgb_preds))
max_val = max(max(y_test), max(xgb_preds))
plt.plot([min_val, max_val], [min_val, max_val], linestyle="--", color="red",␣
 ↪label="Ideal Fit (y = x)")
```

```python
# Labels and title
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("XGBoost: Actual vs. Predicted Price")
plt.legend()
plt.show()
```



[ ]: