Real Estate Price Prediction
Building a Regression Model to Predict Apartment Prices Based on Features

Author: Katarzyna Brzeski
4th-year Informatics student, AI & Data Science specialization
Date: 2025

1. Problem Description

Real estate price prediction is a key issue in data analysis and machine learning.
The goal of the project is to build a regression model that, based on specific features of a property (e.g. location, area, number of rooms), can accurately predict its price.
Such models are widely used in the real estate industry to help buyers and sellers make better financial decisions.

Choice of Programming Language – Justification

The entire project was developed using Python and the Jupyter Notebook environment. These tools were chosen for their versatility and convenience in data analysis, their rich visualization capabilities, ease of building predictive models, and seamless integration with API interfaces.

Jupyter Notebook enabled fast, interactive code iteration, which greatly facilitated the process of data exploration, visualization, and model testing. It provided a clear, step-by-step workflow ideal for analytical projects.

Python was the core language thanks to its powerful ecosystem of libraries such as:

- **pandas** – for data manipulation and preprocessing,
- **matplotlib / seaborn** – for data visualization,
- **scikit-learn** – for machine learning models and evaluation,
- **xgboost** – for advanced boosting models,
- **flask** – for deploying a simple prediction API.

This stack allowed for an end-to-end solution covering every stage of the project — from data analysis and feature engineering, through model training and evaluation, to deployment via an API. The choice of Python and Jupyter was crucial in ensuring efficiency, flexibility, and reproducibility of the entire machine learning pipeline.

2. Algorithm Description and Justification

To build the regression model, I selected three algorithms:

- **Linear Regression** was used as a baseline model — it is fast, simple, and easy to interpret.
- **Random Forest** is a non-linear, ensemble-based model built on decision trees. It handles interactions and non-linearities well.
- **XGBoost** is an advanced boosting technique that uses iterative learning and error optimization. It is known for high performance in regression tasks.

These algorithms vary in complexity, allowing for a meaningful comparison between simpler and more advanced approaches.
The best-performing model will be selected based on evaluation metrics such as **Mean Squared Error (MSE)** and **R^2 score**.
Using multiple models helps identify which one generalizes best and balances accuracy with interpretability.

3. Regression Model Comparison

Feature	Linear Regression	Random Forest	XGBoost
Model Type	Linear model	Ensemble (bagging)	Ensemble (boosting)
Complexity	Low	Medium	High
Overfitting Resistance	Low (on complex data)	High	Very high (with regularization)
Training Time	Very fast	Medium	Longer (depends on parameters)
Interpretability	Very good	Limited (many trees)	Low (many iterations)
Nonlinearity Handling	Poor	Very good	Very good
Missing Data Handling	Needs imputation/removal	Partially tolerant	Automatically supported
Prediction Type	Simple average	Mean of tree predictions	Cumulative error correction

4. Dataset Description

The dataset comes from Kaggle and contains real estate listings in New York City.
It includes 4,801 entries covering the years 2023–2024.

Key features in the dataset:

- `BROKERTITLE`: Broker name
- `TYPE`: Property type
- `PRICE`: Price
- `BEDS`: Number of bedrooms
- `BATH`: Number of bathrooms
- `PROPERTYSQFT`: Square footage
- `ADDRESS`: Full address
- `STATE`: State
- `LOCALITY`: City
- `SUBLOCALITY`: District
- `STREET_NAME`, `LATITUDE`, `LONGITUDE`: Geolocation data
- `FORMATTED_ADDRESS`: Formatted text address

5. Data Preprocessing

Data preprocessing was adjusted to meet the requirements of the regression models:

- Outliers were removed using the 1.5 * IQR rule.
- Categorical variables were encoded using One-Hot Encoding.
- The target variable (price) was log-transformed to normalize its distribution.
- Feature engineering was applied to create new variables such as:
    - `LUXURY_HOME` (binary label for high-end properties)
    - `PRICE_PER_SQFT` (price per square foot)

Loading the data and its initial analysis.

Basic information about the dataset and basic descriptive statistics.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4801 entries, 0 to 4800
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   BROKERTITLE                  4801 non-null   object
 1   TYPE                         4801 non-null   object
 2   PRICE                        4801 non-null   int64
 3   BEDS                         4801 non-null   int64
 4   BATH                         4801 non-null   float64
 5   PROPERTYSQFT                 4801 non-null   float64
 6   ADDRESS                      4801 non-null   object
 7   STATE                        4801 non-null   object
 8   MAIN_ADDRESS                 4801 non-null   object
 9   ADMINISTRATIVE_AREA_LEVEL_2  4801 non-null   object
 10  LOCALITY                     4801 non-null   object
 11  SUBLOCALITY                  4801 non-null   object
 12  STREET_NAME                  4801 non-null   object
 13  LONG_NAME                    4801 non-null   object
 14  FORMATTED_ADDRESS            4801 non-null   object
 15  LATITUDE                     4801 non-null   float64
 16  LONGITUDE                    4801 non-null   float64
dtypes: float64(4), int64(2), object(11)
memory usage: 637.8+ KB
None
```

	PRICE	BEDS	BATH	PROPERTYSQFT	LATITUDE	LONGITUDE
count	4.801000e+03	4801.000000	4801.000000	4801.000000	4801.000000	4801.000000
mean	2.356940e+06	3.356801	2.373861	2184.207862	40.714227	-73.941601
std	3.135525e+07	2.602315	1.946962	2377.140894	0.087676	0.101082
min	2.494000e+03	1.000000	0.000000	230.000000	40.499546	-74.253033
25%	4.990000e+05	2.000000	1.000000	1200.000000	40.639375	-73.987143
50%	8.250000e+05	3.000000	2.000000	2184.207862	40.726749	-73.949189
75%	1.495000e+06	4.000000	3.000000	2184.207862	40.771923	-73.870638
max	2.147484e+09	50.000000	50.000000	65535.000000	40.912729	-73.702450

ANALYSIS

1. Dataset structure

The dataset contains 4,801 rows and 17 columns.

All columns are complete, meaning there are no NULL values (Non-Null Count equals the number of rows).

The data types include both numerical variables (int64, float64) and text variables (object).

Descriptive statistics

Property prices (PRICE): The average price is 2,356,940 USD.

Extreme values: The minimum is 2,494 USD, and the maximum is 2,147,483,647 USD. This may indicate potential outliers that should be examined in more detail.

Number of bedrooms (BEDS): The average number of bedrooms is 3.36, with a minimum of 1 and a maximum of 50 — which may also represent an anomaly.

Number of bathrooms (BATH): The average is 2.37, ranging from 0 to 50, which may also be unusual.

Property size (PROPERTYSQFT): The average area is 2,184 square feet.
Range: From 230 to 65,535 square feet, which may indicate both small and very luxurious properties.

Missing data check

```
missing_values = df.isnull().sum()
print(missing_values)
```

```
BROKERTITLE                  0
TYPE                         0
PRICE                        0
BEDS                         0
BATH                         0
PROPERTYSQFT                 0
ADDRESS                      0
STATE                        0
MAIN_ADDRESS                 0
ADMINISTRATIVE_AREA_LEVEL_2  0
LOCALITY                     0
SUBLOCALITY                  0
STREET_NAME                  0
LONG_NAME                    0
FORMATTED_ADDRESS            0
LATITUDE                     0
LONGITUDE                    0
dtype: int64
```

Required columns (features for the ML model):

PRICE – our target variable.

BEDS – number of bedrooms.

BATH – number of bathrooms.

PROPERTYSQFT – property size (in square feet).

LATITUDE / LONGITUDE – geographic coordinates.

LOCALITY – neighborhood (to be converted using One-Hot Encoding).

TYPE – may have a significant impact on the price.

Text data not related to prediction:

BROKERTITLE – Information about the real estate agent; does not affect the property price.

ADDRESS / MAIN_ADDRESS / FORMATTED_ADDRESS – Full address fields, redundant due to LATITUDE and LONGITUDE.

STATE – All properties are located in New York; not useful for the model.

ADMINISTRATIVE_AREA_LEVEL_2 – Same as LOCALITY but in a different format. Redundant.

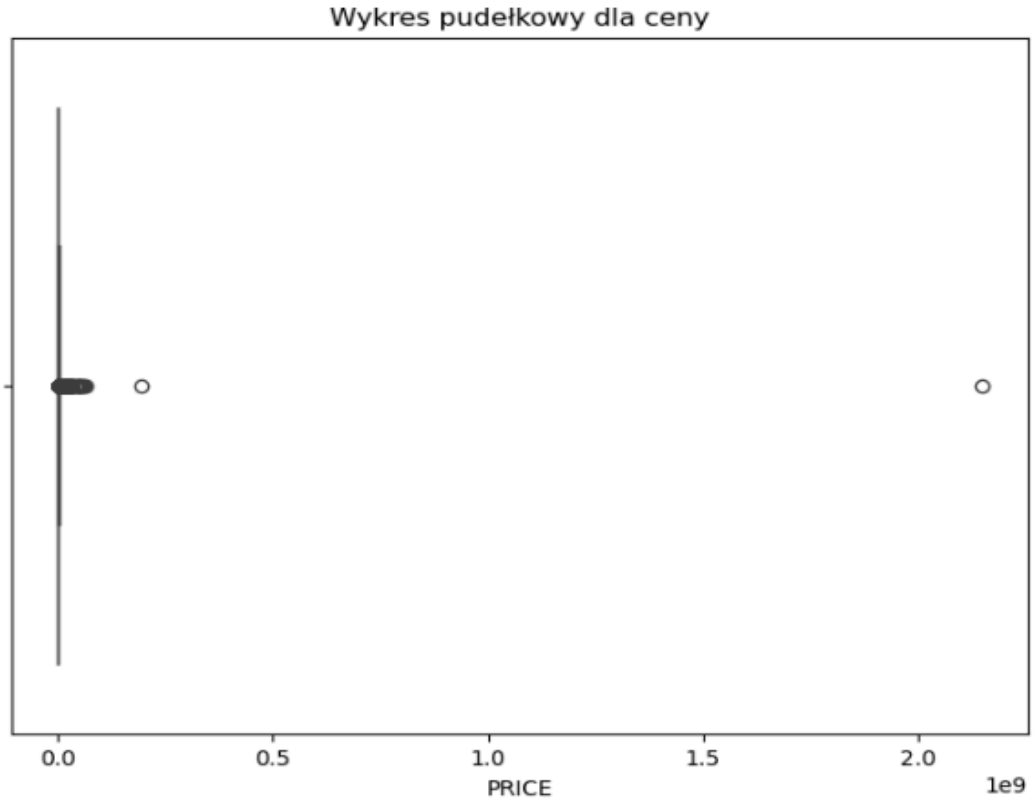
SUBLOCALITY – Often overlaps with LOCALITY.

STREET_NAME / LONG_NAME – Very detailed data that adds little value; a high number of unique values makes modeling difficult.

One-Hot Encoding (OHE) for categorical variables **LOCALITY** and **TYPE** creates new columns for each category and marks them with 0 or 1.

Outlier identification

Analysis and cleaning of the **PRICE** column, which is the target variable.



Analysis

The boxplot for the **PRICE** variable shows the distribution of property prices.There is a clear concentration of data in the lower part of the price range, as well as the presence of outliers at the upper end of the distribution.

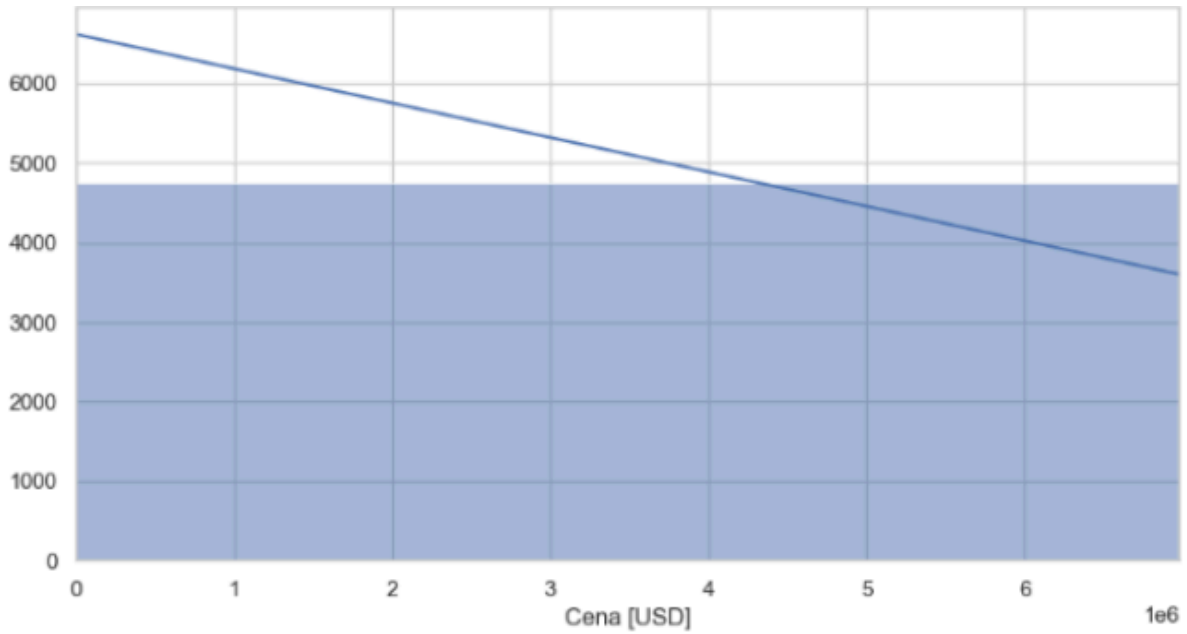
Application of the 1.5 * IQR rule

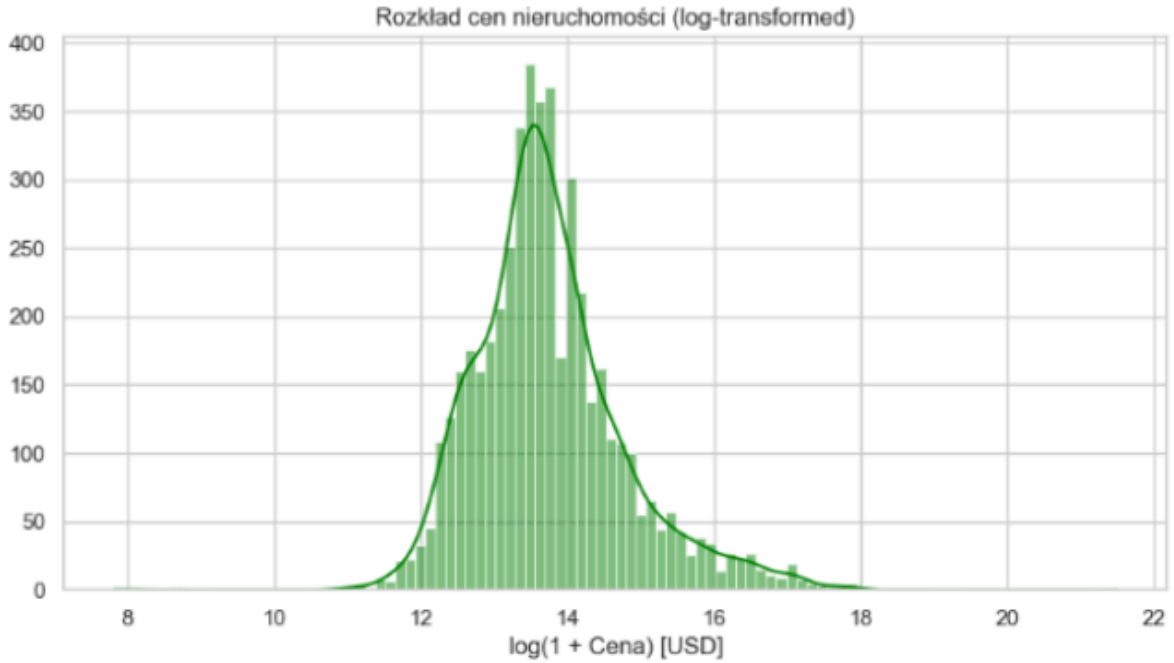
```
Q1 = df_encoded['PRICE'].quantile(0.25)
Q3 = df_encoded['PRICE'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = df_encoded[(df_encoded['PRICE'] < lower_bound) | (df_encoded['PRICE'] > upper_bound)]
print(f"Liczba outlierów w cenach: {len(outliers)}")
```

```
Liczba outlierów w cenach: 559
```

559 observations fall outside the typical data range.

Logarithmic transformation applied to PRICE, which "flattens" the distribution and reduces the impact of extreme values.





Analysis

Original price distribution

The Property Price Distribution (original) plot shows a highly skewed distribution.

Most properties fall within the lower price range, while a few outliers appear at extremely high prices.

Price distribution after logarithmic transformation

The Property Price Distribution (log-transformed) plot reveals that after applying the logarithmic transformation, the data becomes much more symmetric and closer to a normal distribution.

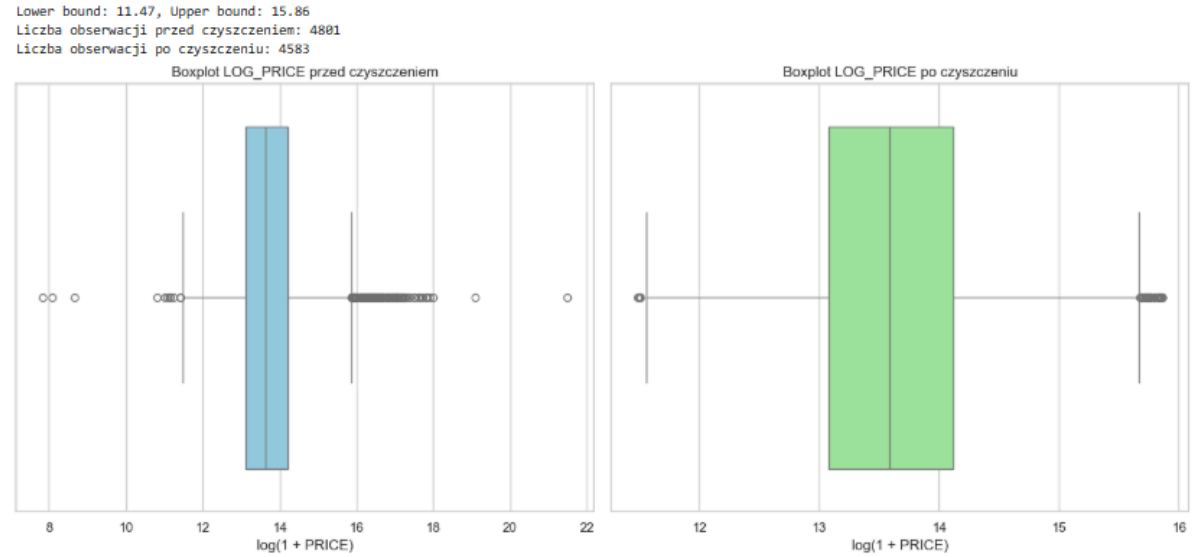
Benefits of transformation

- Reduces the impact of outliers – extreme values are compressed, which limits their influence on the model.

Checking extreme values before and after transformation.

```
Min cena: 2494
Max cena: 2147483647
99 percentyl: 22500000.0

Min log-cena: 7.822044008185619
Max log-cena: 21.487562597358306
```

This indicates that there are a few extremely high values that may affect the modeling process.

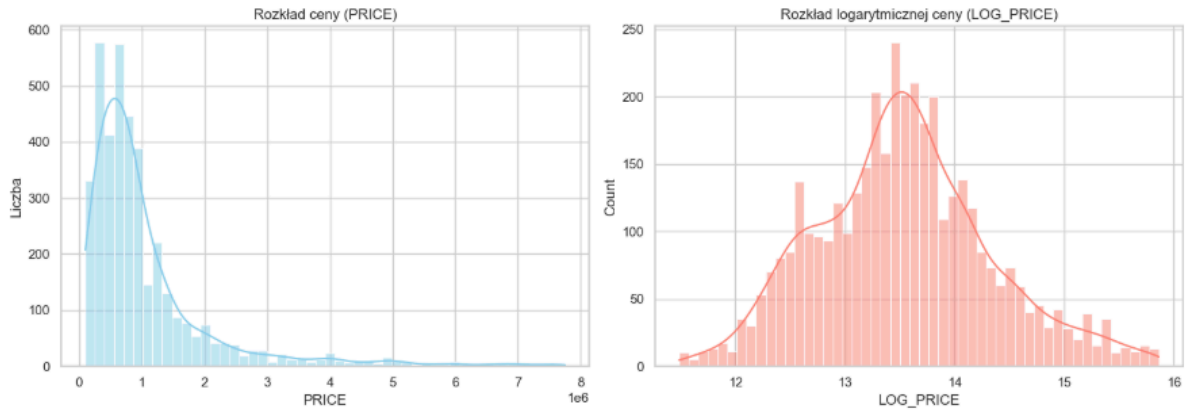


218 observations were removed, which accounts for approximately 4.5% of the data.
This shows that outliers were relatively rare, but their removal improves the quality of the dataset.

Before cleaning: The distribution of LOG_PRICE included outliers beyond the upper and lower limits, which affected the median and intervals.

After cleaning: The median remains nearly unchanged, indicating that the central values of the data were preserved.
The range between the lower (Q1) and upper quartiles (Q3) is now more stable, meaning the dataset is more representative of the majority of properties.

Histogram: PRICE vs LOG_PRICE



PRICE distribution:
The histogram of PRICE is highly right-skewed, indicating that most property prices are concentrated around lower values.
The peak of the distribution appears around 500,000, and the number of properties priced above 1 million is relatively small.
The distribution is irregular, which makes accurate predictive modeling difficult and may lead to overfitting if the original data is used as-is.

After logarithmic transformation:
The distribution of LOG_PRICE becomes more similar to a normal distribution.
There is a clear peak around the value of 14, meaning that most prices fall within a specific logarithmic range.
The logarithmic transformation helps normalize the data by reducing the influence of very high or very low property prices.

Outlier detection for BEDS and BATH – Analysis

BEDS

- Typical range: 2–4 bedrooms
- Outliers above 7: 199 observations (~4.3% of the dataset)
- No outliers below the lower limit (−1)
- Values above 6 may indicate luxury properties, multi-family homes, villas, or hotels.

BATH

- Typical range: 1–3 bathrooms
- Outliers above 6: 70 observations (less than 2%)
- No lower outliers
- As with bedrooms, high values may be valid but rare.

Descriptive statistics (count, mean, std, min, max, quartiles)

```
=== Statystyki opisowe: BEDS ===
count    4583.000000
mean        3.256164
std         2.432918
min         1.000000
25%         2.000000
50%         3.000000
75%         4.000000
max        40.000000
Name: BEDS, dtype: float64
```

Value distribution – Bedrooms:
Most properties have between 1–4 bedrooms:

- 1 bedroom: 817 cases
- 2 bedrooms: 987 cases
- 3 bedrooms: 1,421 cases (most frequent)

Outliers above the 95th percentile: values greater than 7 bedrooms
There are 199 outlier values, including extreme cases such as properties with 30, 32, 35, 36, and even 40 bedrooms.

```
=== Statystyki opisowe: BATH ===
count    4583.000000
mean        2.241589
std         1.589526
min         0.000000
25%         1.000000
50%         2.000000
75%         3.000000
max        32.000000
Name: BATH, dtype: float64
```

Value distribution – Bathrooms:
Typical properties have 1–3 bathrooms:

- 1 bathroom: 1,512 cases
- 2 bathrooms: 1,661 cases
- 3 bathrooms: 461 cases

Outliers above the 95th percentile: values greater than 6 bathrooms
There are 70 outlier values, including extreme cases such as 16, 20, 24, and 32 bathrooms.

Outlier cleaning using the 1.5 * IQR method.

BEDS Analysis

Before cleaning

- Observations: The distribution had many outliers, with the maximum number of bedrooms reaching as high as 40.
- Most data fell within the range of 1 to 5, clearly reflected in the compact quartiles (Q1 = 2, Q3 = 4).
- Extreme values above 7 bedrooms were far beyond the upper IQR limit and could distort the analysis.

After cleaning (1.5 * IQR)

- Data range: Values were limited to the range of 1–7 bedrooms.
- The maximum number of bedrooms is now 7, eliminating the influence of extreme outliers.
- Distribution change: The distribution became more compact and representative of typical properties.
- Outliers, likely related to unusual or luxury properties, were removed.

BATH Analysis

Before cleaning

- Data distribution: Most values were within the range of 1–3 bathrooms, which matched the main quartiles:
    - Q1 = 1.0
    - Q3 = 3.0
- A large number of outliers appeared above the IQR upper limit (6), visible as many points beyond the upper whisker.
- The maximum value was 32, a significant outlier likely referring to unusual properties (e.g. apartment buildings or luxury estates).

After cleaning

- Data distribution: Values were restricted to the range of 0 to 6 bathrooms (upper IQR limit = 6.0).
- Extreme values such as 16, 20, 24, and 32 bathrooms were removed.
- The distribution became more compact, facilitating further analysis and modeling.

Outlier detection for PROPERTYSQFT

```
=== Statystyki opisowe: PROPERTYSQFT ===
count     4251.000000
mean      1844.586621
std       1631.994812
min        246.000000
25%       1112.000000
50%       2082.000000
75%       2184.207862
max      55300.000000
Name: PROPERTYSQFT, dtype: float64
```

Average area: 1,844.59 square feet, indicating moderately sized properties.
Median (50%): 2,082 square feet, suggesting that half of the properties are smaller than this value, and the other half are larger.
Standard deviation: 1,632 — relatively high, indicating considerable variability in property size.

Interquartile range (IQR):

- **Q1 (25%):** 1,112 square feet
- **Q3 (75%):** 2,184.21 square feet
- **IQR ≈ 1,072.21**, which describes the concentration of data within typical values.

Maximum value: 55,300 square feet — an extreme outlier, likely referring to a luxury or commercial property.

Top 10 extreme properties

These range from 6,000 up to 55,300 square feet, indicating highly atypical cases.

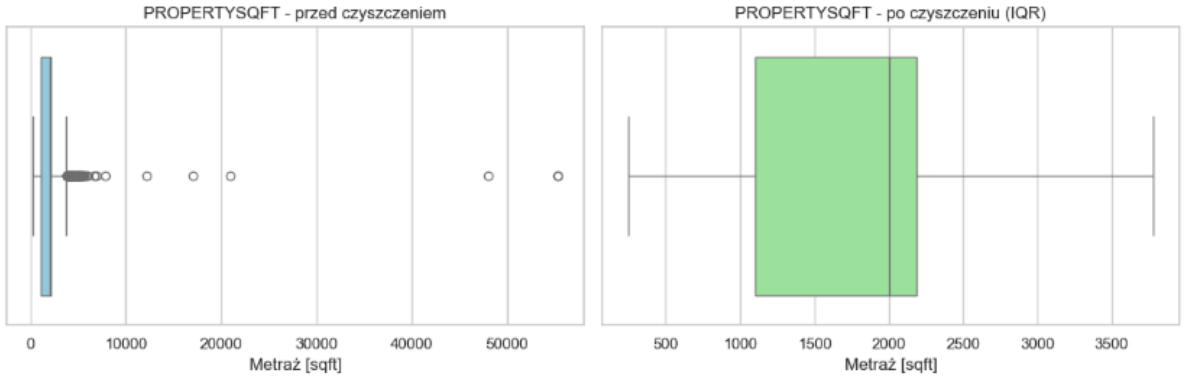
Examples: 21,000, 48,000, and 55,300 square feet.

Such values may represent data entry errors or specific cases (e.g. shopping centers, apartments).

```
Największe wartości (TOP 10):
2146    55300.0
2148    55300.0
823     48000.0
2054    21000.0
4353    17000.0
1295    12200.0
4126     7868.0
3403     6862.0
1411     6707.0
2606     6000.0
Name: PROPERTYSQFT, dtype: float64
```

1.5 * IQR

```
IQR = 1072.2078620000002
Lower bound = -496.3117930000003, Upper bound = 3792.5196550000005
Usunięto 96 obserwacji
```



Analysis

Before cleaning
Data distribution:
Numerous outliers are visible, significantly exceeding the upper whisker of the boxplot (upper bound = 3,792.52 sqft).
The maximum value reaches 55,300 sqft, indicating extreme cases likely related to large properties or data errors.

Effect on descriptive statistics:
Outliers considerably increase the mean (1,844.59 sqft) and standard deviation (1,632 sqft), which may distort the overall representativeness of the data.

After cleaning
Distribution changes:
The data is now limited to the range from 246 sqft (min) to 3,792.52 sqft (upper bound).
96 observations were removed, improving the consistency of the distribution and reducing the impact of extreme values.
The new maximum value is 3,792.52 sqft, which is more representative of typical properties.
The mean, median, and standard deviation are now more aligned with the actual distribution of the data.

Outlier detection for LATITUDE and LONGITUDE

LATITUDE column
The data is uniform; no outliers are present, suggesting that this column does not require cleaning.

LONGITUDE column
There are 109 outlier values below the lower bound. These may represent properties located on city borders, islands, or areas outside of New York City.
Observations with LONGITUDE < -74.1717 are removed.

After cleaning LONGITUDE
Distribution changes:
The lower outlier values (109 observations below −74.1717) were removed, resulting in a more compact data distribution.
The interquartile range (IQR) remains unchanged, indicating that the typical longitude values were not affected.

Values in the range **−74.1717 to −73.6772** are now more representative of properties located in New York City.

Improved data quality:
Removing 109 observations makes the dataset better suited for analyzing New York City real estate.

Elimination of outlier locations:
The removed values likely corresponded to properties outside the city or data entry errors.

No impact on upper values:
Since there were no outliers above the upper bound, the data in that range remained unchanged.

Analysis of LOCALITY and TYPE

```
=== Lokalizacje (LOCALITY) ===
LOCALITY_Brooklyn: 6 obserwacji
LOCALITY_Flatbush: 1 obserwacji
LOCALITY_Kings County: 451 obserwacji
LOCALITY_New York: 1848 obserwacji
LOCALITY_New York County: 910 obserwacji
LOCALITY_Queens: 6 obserwacji
LOCALITY_Queens County: 556 obserwacji
LOCALITY_Richmond County: 55 obserwacji
LOCALITY_The Bronx: 5 obserwacji
LOCALITY_United States: 30 obserwacji
```

```
=== Typy nieruchomości (TYPE) ===
TYPE_Coming Soon: 2 obserwacji
TYPE_Condo for sale: 815 obserwacji
TYPE_Condop for sale: 5 obserwacji
TYPE_Contingent: 68 obserwacji
TYPE_For sale: 14 obserwacji
TYPE_Foreclosure: 13 obserwacji
TYPE_House for sale: 863 obserwacji
TYPE_Land for sale: 46 obserwacji
TYPE_Mobile house for sale: 1 obserwacji
TYPE_Multi-family home for sale: 447 obserwacji
TYPE_Pending: 212 obserwacji
TYPE_Townhouse for sale: 141 obserwacji
```

Based on these results, we have a clear picture: some categories are strongly represented, but many of them are rare values that may introduce noise into the model.

Analysis of property types (TYPE):

Dominant types:

- **House for sale:** 863 observations – the most common property type, suggesting a large number of single-family homes in the dataset.
- **Condo for sale:** 815 observations – the second most frequent type, indicating the popularity of apartments for sale in New York City.
- **Multi-family home for sale:** 447 observations – shows a significant presence of multi-family properties.

Rare types:

- **Mobile house for sale:** Only 1 observation – likely a non-standard property type for New York City.
- **Condop for sale:** 5 observations – represents a specific type of property available in limited cases.

Unusual types:

- **Foreclosure (13 observations)** and **Contingent (68 observations):** May refer to properties in special legal situations, such as bank repossessions or pending sales.

- **Land for sale (46 observations):** Land sales are relatively rare in New York City, which aligns with expectations.

Analysis of location (LOCALITY)

Location distribution

Dominant locations:

- **New York:** 1,848 observations – the largest portion of the dataset, likely representing properties concentrated in Manhattan.

- **New York County:** 910 observations – overlaps with the Manhattan area, confirming its central role in the dataset.

- **Queens County:** 556 observations – indicates a significant number of properties in this borough, though fewer than in New York County.

- **Kings County:** 451 observations – includes Brooklyn, highlighting its importance as a property location.

Rare locations:

- **Brooklyn** and **Queens:** Only 6 observations – possibly due to specific data formatting or limited representation.

- **Flatbush:** Just 1 observation – likely a result of incomplete or inconsistent labeling for this location.

Unusual locations:

- **United States:** 30 observations – likely misclassified locations or properties outside New York City.

- **Richmond County** (55 observations) and **The Bronx** (5 observations) – indicate a smaller representation of properties from these areas.

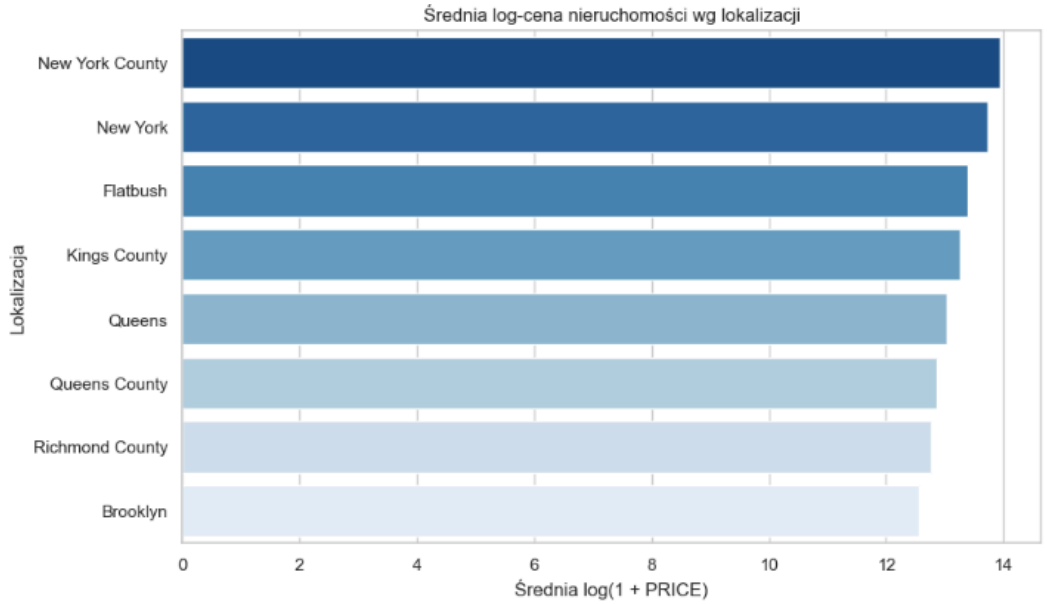
Data cleaning decisions:

- **United States (30 observations):** Removed – likely misclassified due to incorrect or missing geolocation data.

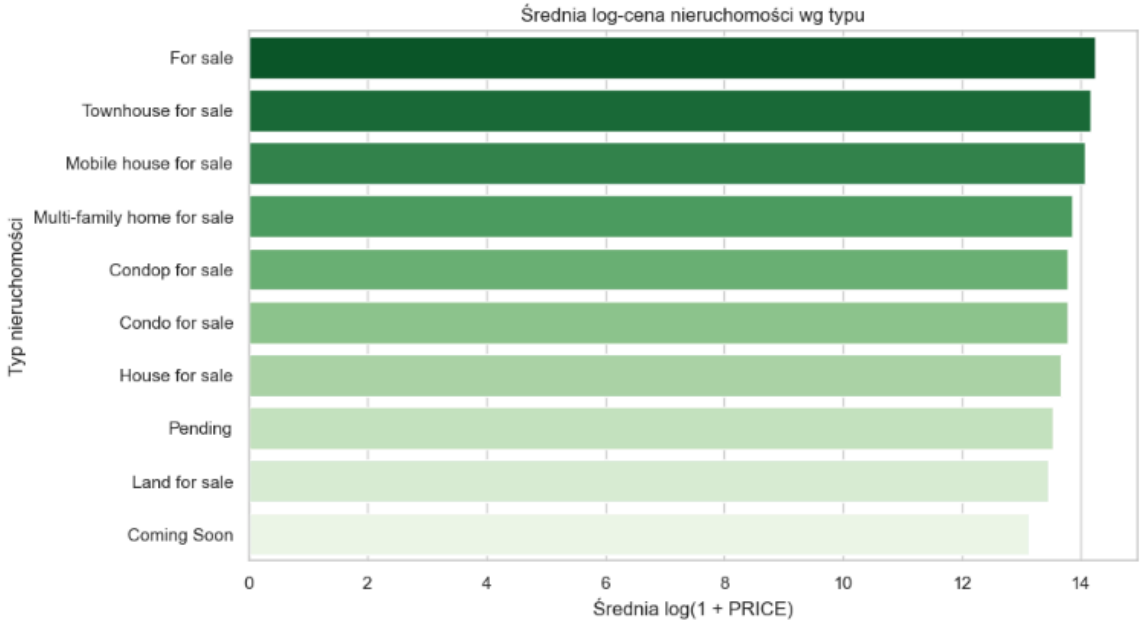
- **The Bronx (5 observations):** Removed – the group is too small for the model to learn meaningful patterns.

- **Foreclosure (13)** and **Contingent (68):** Removed – these indicate the legal status of the offer, not the property type.

Analysis of the impact of location and property type



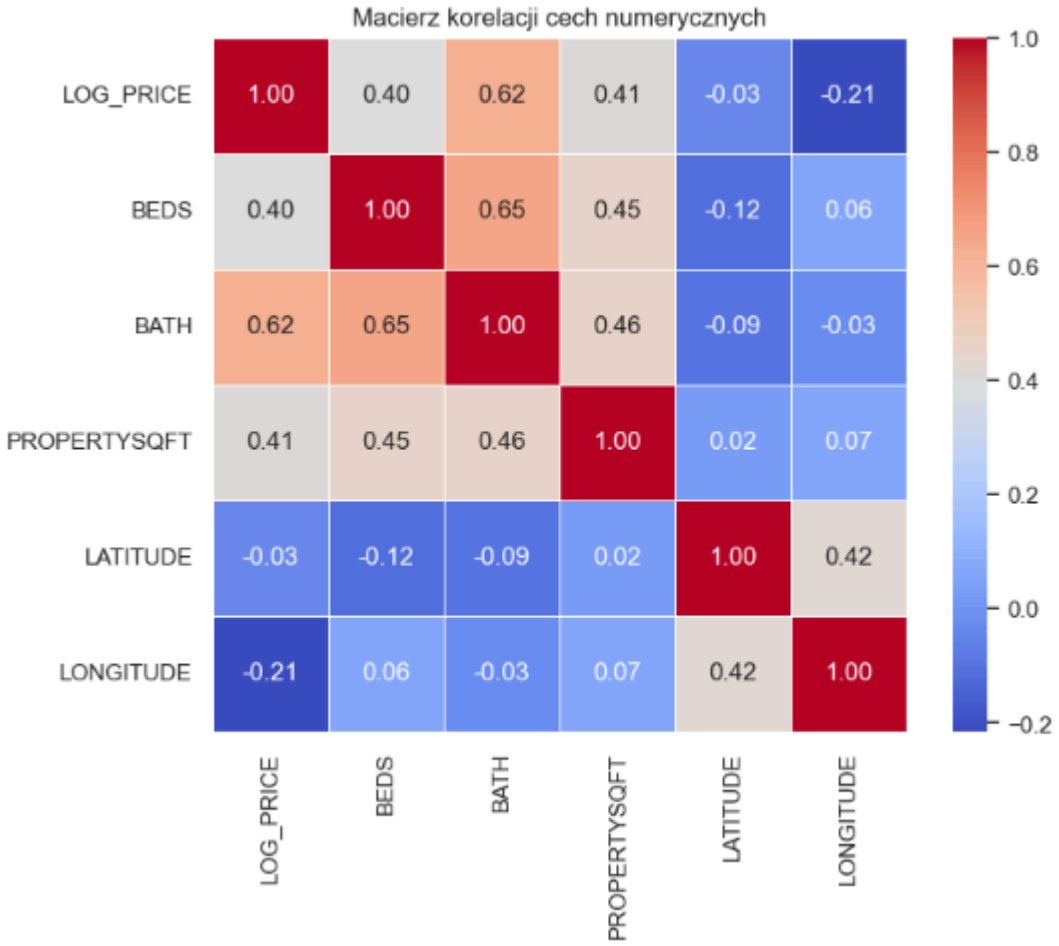
The chart shows the average logarithmic property price (*log(1 + PRICE)*) for different locations.



Property type clearly influences prices.

Luxury listings (e.g., Townhouse and For sale) are significantly more expensive than basic options.

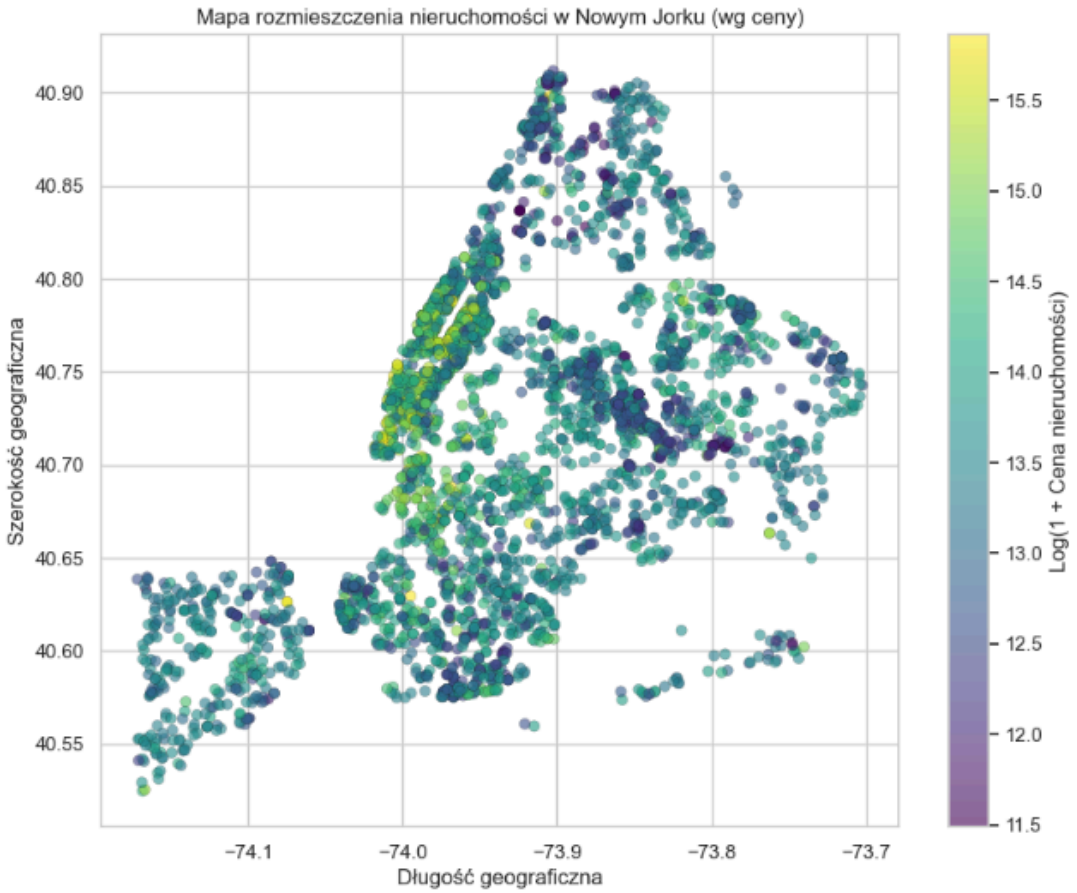
The correlation matrix for the features 'LOG_PRICE', 'BEDS', 'BATH', 'PROPERTYSQFT', 'LATITUDE', and 'LONGITUDE' allows us to evaluate the relationships between the selected features.



Correlation with LOG_PRICE

- BATH (0.62):
  Strong positive correlation. The number of bathrooms has the greatest impact on property price, likely due to the prestige and comfort of larger homes.
- PROPERTYSQFT (0.41):
  Moderate positive correlation. Property size naturally affects the price, but its influence is lower than that of the number of bathrooms.
- BEDS (0.40):
  Moderate positive correlation. The number of bedrooms also influences the price, though to a lesser extent than bathrooms.
- LATITUDE (−0.03):
  Very weak correlation. Latitude has little to no meaningful impact on property price.
- LONGITUDE (−0.21):
  Weak negative correlation. Longitude may suggest a tendency for lower prices in the western areas of the city.
- BEDS and BATH (0.65):
  Strong correlation between the number of bedrooms and bathrooms.
  Larger properties typically have more of both.

Property distribution map by price



General conclusions

- Impact of location: Properties located closer to the city center clearly have higher prices. Log-prices in the range of 15.0–15.5 are especially visible in the most exclusive areas of Manhattan.
- Price variation: Areas with the lowest logarithmic prices likely represent more residential or peripheral zones, which aligns with expectations.

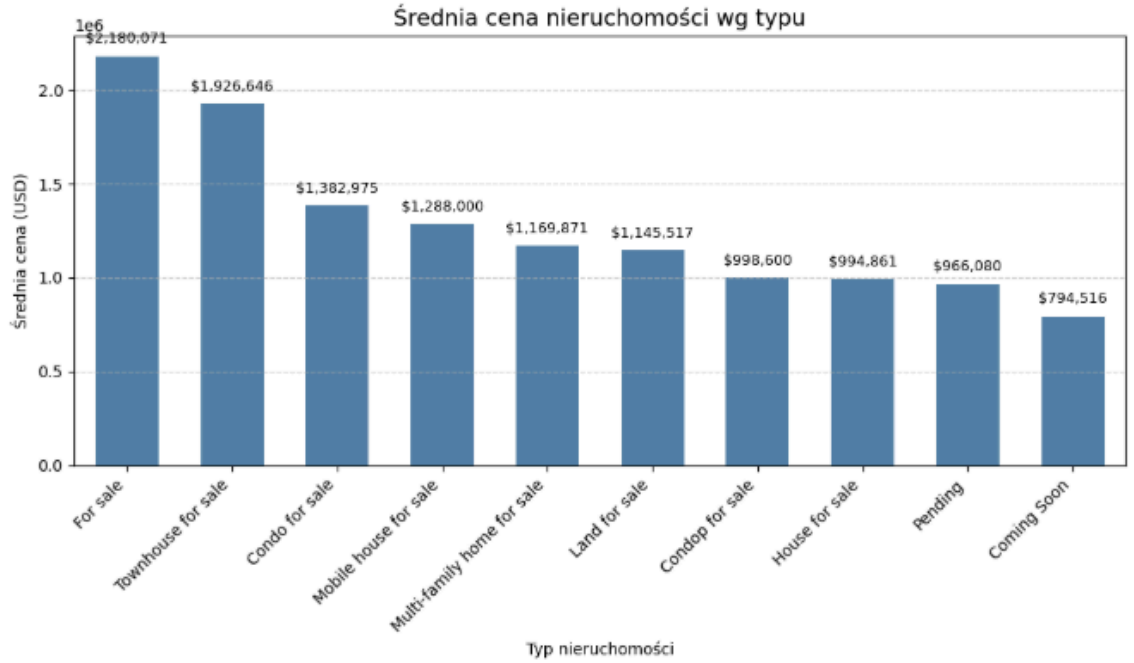
Location is a key factor affecting price — the closer to the center, the higher the logarithmic property prices.
The price distribution is consistent with the geographic structure of New York City — exclusive central areas vs. more affordable outskirts.

Feature engineering

Adding the LUXURY_HOME feature allows the model to:

- More easily distinguish luxury/high-priced properties from standard ones,
- Capture non-linear differences in pricing,
- Potentially improve prediction accuracy, especially for very expensive homes.

Analysis of average price by property type
By examining which property types have the highest prices,
we can better define what “luxury” means
and later justify the creation of the LUXURY_HOME feature.



TYPE	PRICE	LOG_PRICE
For sale	2180071.43	14.24
Townhouse for sale	1926646.41	14.16
Mobile house for sale	1288000.00	14.07
Multi-family home for sale	1169870.86	13.87
Condop for sale	998600.00	13.79
Condo for sale	1382975.00	13.78
House for sale	994860.68	13.66
Pending	966079.83	13.54
Land for sale	1145516.62	13.46
Coming Soon	794515.81	13.13

LUXURY_HOME based on multiple conditions makes the feature more intelligent and context-aware, rather than relying solely on price.

Defining thresholds for a luxury home:
A property is considered **LUXURY_HOME = 1** if it meets **all** of the following:

- **PRICE > 1,500,000 USD**
- **PROPERTYSQFT > 2,500 sq ft**
- **BATH ≥ 3 bathrooms**
- **BEDS ≥ 4 bedrooms**
- **TYPE** belongs to luxury categories (e.g. *For sale*, *Townhouse*, *Mobile house*)

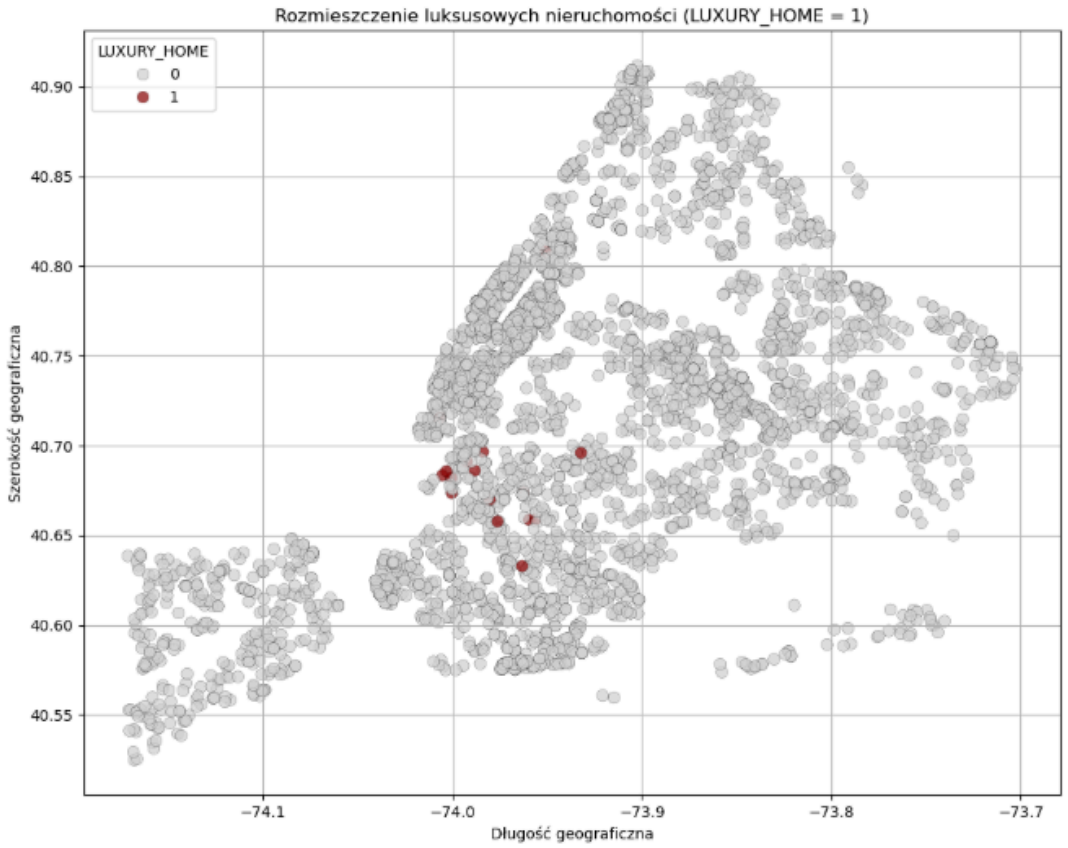
Otherwise: **LUXURY_HOME = 0**

This compound feature helps the model capture the unique characteristics of high-end real estate more effectively.

0nly about **0.46%** of the properties meet the multi-criteria luxury conditions (**LUXURY_HOME = 1**).

Luxury properties are naturally rare, and this logic reflects that well.

Where are luxury properties located on the map?



The scatter plot shows the distribution of luxury properties (**LUXURY_HOME = 1**) compared to regular properties (**LUXURY_HOME = 0**).

Luxury properties, marked in **red**, are clearly concentrated in specific areas, highlighting high-value zones in the real estate market.

In contrast, regular properties, shown in **gray**, are more widely dispersed across the map.

The axes represent **latitude (LATITUDE)** and **longitude (LONGITUDE)**.

This visualization reveals clusters of luxury homes, indicating their preferred locations.

Comparison: Luxury Properties vs. the Rest

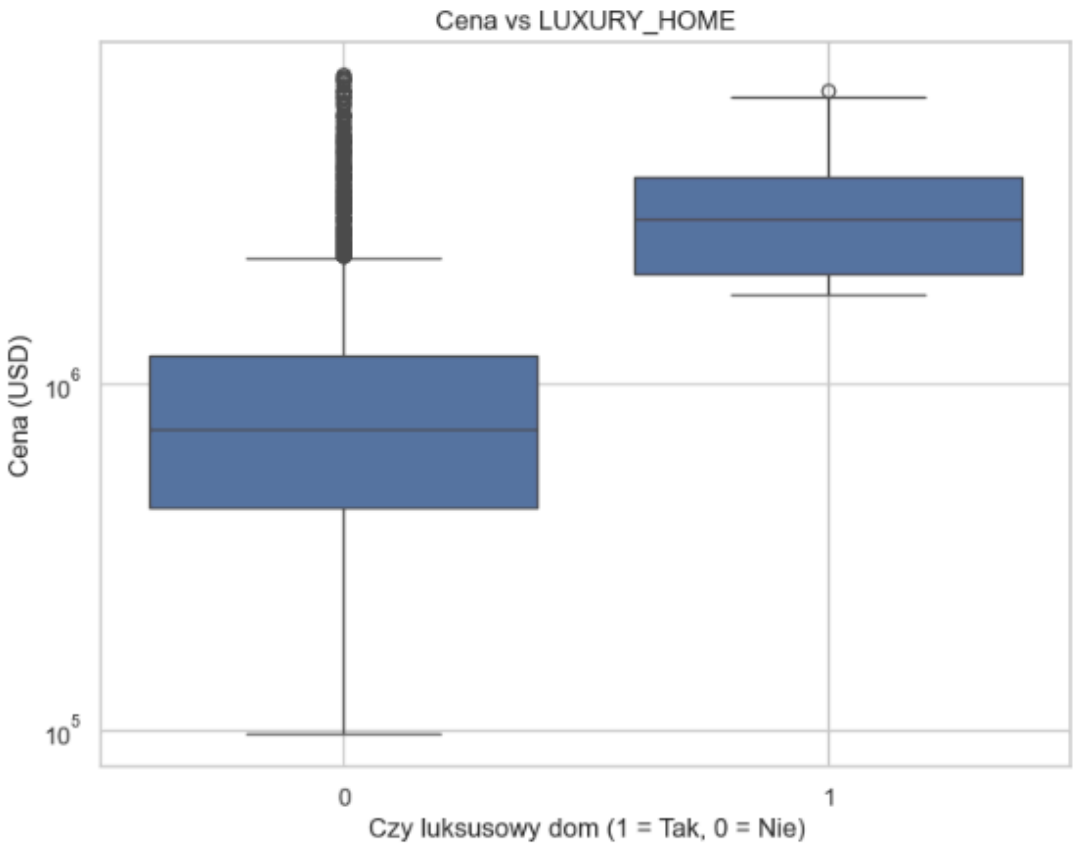
```
df_encoded.groupby('LUXURY_HOME')[['PRICE', 'PROPERTYSQFT', 'BATH', 'BEDS']].mean().round(0)
```

LUXURY_HOME	PRICE	PROPERTYSQFT	BATH	BEDS
0	1049918.0	1725.0	2.0	3.0
1	3433278.0	3168.0	4.0	5.0

The results show that the variable LUXURY_HOME works perfectly, clearly separating luxury properties from the rest.

Conclusions
The differences are significant, distinct, and meaningful — luxury homes have an average price more than 3 times higher than non-luxury properties.
The LUXURY_HOME feature is logical, robust, and useful.

Impact of LUXURY_HOME on PRICE



Conclusions
Luxury homes have clearly higher prices.
The price distribution is less varied, indicating a more uniform market value within this category.
The median price for luxury homes is significantly higher than for non-luxury properties.

Non-luxury homes show a wider price range, suggesting greater diversity within this group.
There are more outliers, which may indicate the presence of lower-tier properties with unusually high prices.
The LUXURY_HOME variable has a significant impact on price variability.

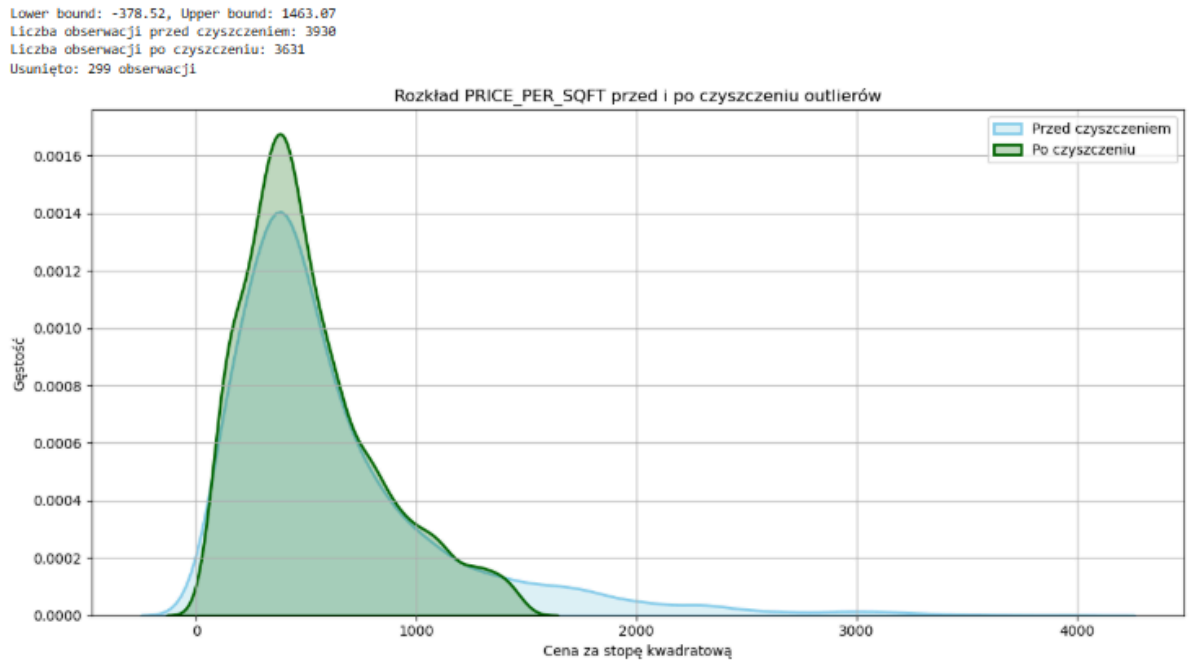
Adding the feature PRICE_PER_SQFT (price per square foot) is one of the most important metrics in real estate analysis:

- It allows standardized comparison of properties of different sizes.
- It helps identify locations where the price per square meter is particularly high.

Checking basic statistics for PRICE_PER_SQFT

Outlier removal

Distribution of PRICE_PER_SQFT before and after outlier removal



Analysis

Distribution changes

Before cleaning (blue distribution):

- The distribution was noticeably **stretched by extreme values**.
- A large number of observations with **very high price per square foot** distorted the histogram.
- It was difficult to identify actual data clusters, as typical values appeared **flattened**.

After cleaning (green distribution):

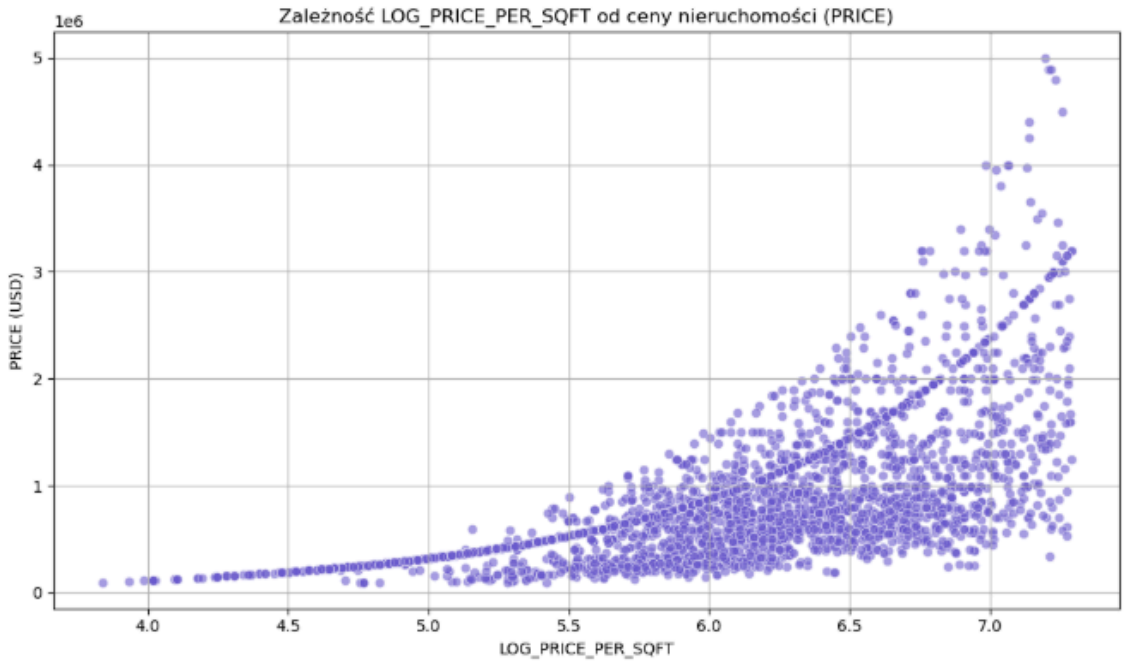
- The distribution became much **more concentrated and natural**.
- Values above **1,463.07 USD** were removed, which accounted for **7.6% of the observations (299 properties)**.
- The main cluster of prices (approximately **312–772 USD**) is now clearly visible.

Creating LOG_PRICE_PER_SQFT — feature transformation

The raw **PRICE_PER_SQFT** variable has a fairly wide range, from approximately **$45 to nearly $4,000**.
Applying a logarithmic transformation **"flattens" these differences**, reduces the influence of extreme values, and **highlights variations within typical ranges**.

Linear regression or XGBoost tend to perform better when the feature distribution is closer to normal.
LOG_PRICE_PER_SQFT often resembles a **bell-shaped (Gaussian) distribution** more closely than the raw values.

Scatter plot: Relationship between LOG_PRICE_PER_SQFT and PRICE




```
correlation = df_encoded_clean[['LOG_PRICE_PER_SQFT', 'PRICE']].corr().iloc[0, 1]
print(f"Korelacja między LOG_PRICE_PER_SQFT a PRICE: {correlation:.2f}")
```

```
Korelacja między LOG_PRICE_PER_SQFT a PRICE: 0.65
```

Does LOG_PRICE_PER_SQFT carry information about PRICE?
The correlation coefficient r = 0.65 indicates a moderate positive relationship.
The scatter plot shows a clear upward trend — the higher the log-transformed price per square foot, the higher the total property price tends to be.

The points become denser and rise to the right, indicating a visible (though nonlinear) relationship — ideal for models like XGBoost or Random Forest.

Should this feature be used in the price prediction model?
Yes — for several reasons:

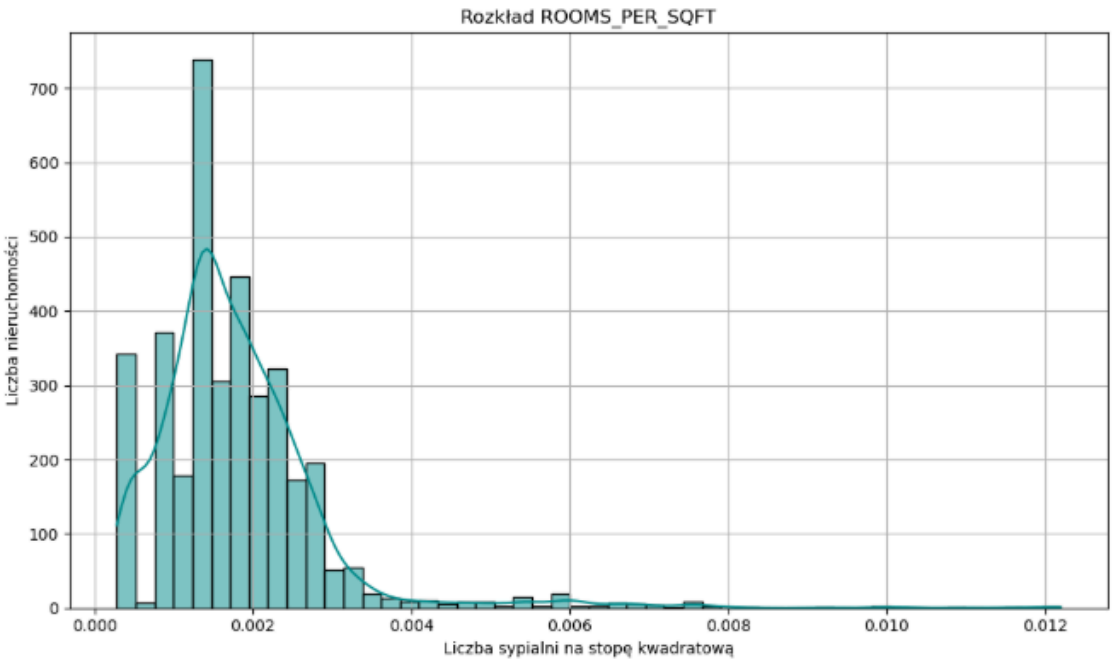
- It has a solid correlation with PRICE
- It captures economic meaning — how much we pay per unit of space
- It complements PROPERTYSQFT — not just “how much space,” but “at what price per unit”

Is the variability of PRICE_PER_SQFT correlated with total property value?
Yes, but moderately — and that’s valuable.

A correlation of 0.65 means that the price per square meter (or foot) does affect the final price, but it’s not the only factor — others like size, property type, and location also play key roles.

New feature: ROOMS_PER_SQFT (bedroom density per square foot)

This new feature, ROOMS_PER_SQFT (or bedroom density per square foot), adds another dimension to interpreting a property's size and internal layout. It can reveal significant differences in apartment or house configurations, even for properties with the same total area. This metric complements existing features such as BATH, PRICE_PER_SQFT, BEDS, and PROPERTYSQFT.



Typical values are concentrated between 0.001 and 0.0025, which aligns perfectly with the previous statistics (25th–75th percentile).
The distribution is highly right-skewed, with a tail extending up to around 0.012, indicating a small number of very "densely packed" properties.

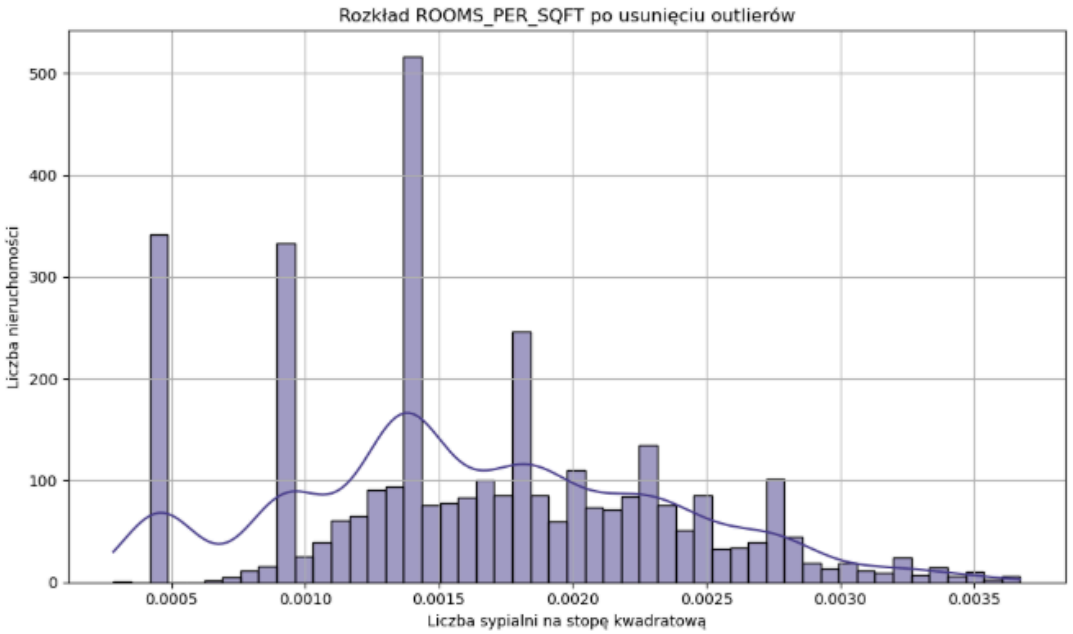
A few properties have unnaturally high ROOMS_PER_SQFT values, possibly due to:

- Very small floor area combined with multiple rooms, or
- Data entry errors, such as underestimated square footage.

Outlier removal using the 1.5 * IQR rule.

Removing outliers from ROOMS_PER_SQFT reduced the dataset from 3,631 to 3,498 observations — only 133 records were removed, which is reasonable and acceptable.

96% of the data was retained, while also eliminating observations that could introduce noise into the model.



Analysis
Distinct spikes (peaks) in the distribution suggest that the data is somewhat grouped or rounded.
Since ROOMS_PER_SQFT is a ratio, even small changes in PROPERTYSQFT with a fixed number of rooms can cause noticeable differences.

Comparison of ROOMS_PER_SQFT between luxury properties (LUXURY_HOME = 1) and standard ones (LUXURY_HOME = 0):
This can reveal a lot about spaciousness and comfort, which are typical characteristics of luxury real estate.

Descriptive statistics by LUXURY_HOME

LUXURY_HOME	count	mean	std	min	25%	50%	75%	max
0	3483.0	0.001641	0.000701	0.000283	0.001183	0.001562	0.002143	0.003672
1	15.0	0.001612	0.000308	0.001104	0.001483	0.001653	0.001749	0.002352

Descriptive statistics by LUXURY_HOME

- Number of observations
    - Standard (0): 3,483
    - Luxury (1): 15
      → Significant imbalance
- Mean
    - Standard: 0.001641
    - Luxury: 0.001612
      → Luxury properties are slightly less dense

- Median (50%)
    - Standard: 0.001562
    - Luxury: 0.001483
      → Confirms more space per bedroom in luxury homes
- 25% – 75% (Interquartile range)
    - Standard: 0.001183 – 0.002143
    - Luxury: 0.001104 – 0.001749
      → Luxury properties show a narrower, lower-density range — more spacious and consistent
- Maximum value
    - Standard: 0.003672
    - Luxury: 0.002352
      → No extremely “crammed” luxury homes

Conclusion:
Luxury properties offer more space per bedroom.
They are not “overloaded” — bedroom counts are better distributed over larger floor areas.
Lower density = greater comfort, prestige, and privacy.

ROOMS_PER_SQFT is a valuable and interpretable feature for the model.

Adding a new feature: LOCATION_CATEGORY based on LATITUDE and LONGITUDE .An effective way to capture the impact of location in the model.

Preview of coordinate values

```
print(df_encoded_clean[['LATITUDE', 'LONGITUDE']].head())
```

```
    LATITUDE  LONGITUDE
0  40.761255 -73.974483
1  40.809448 -73.946777
2  40.615738 -73.969694
3  40.824870 -73.922983
4  40.624996 -74.155306
```

Checking the range of LATITUDE and LONGITUDE

```
print("Zakres LATITUDE:")
print(df_encoded_clean['LATITUDE'].describe())

print("\nZakres LONGITUDE:")
print(df_encoded_clean['LONGITUDE'].describe())
```

```
Zakres LATITUDE:
count    3498.000000
mean       40.716953
std         0.085922
min        40.524763
25%        40.639736
50%        40.726767
75%        40.773155
max        40.911772
Name: LATITUDE, dtype: float64

Zakres LONGITUDE:
count    3498.000000
mean      -73.924990
std         0.095823
min       -74.171463
25%       -73.978178
50%       -73.932469
75%       -73.856165
max       -73.702450
Name: LONGITUDE, dtype: float64
```

Creating LOCATION_CATEGORY

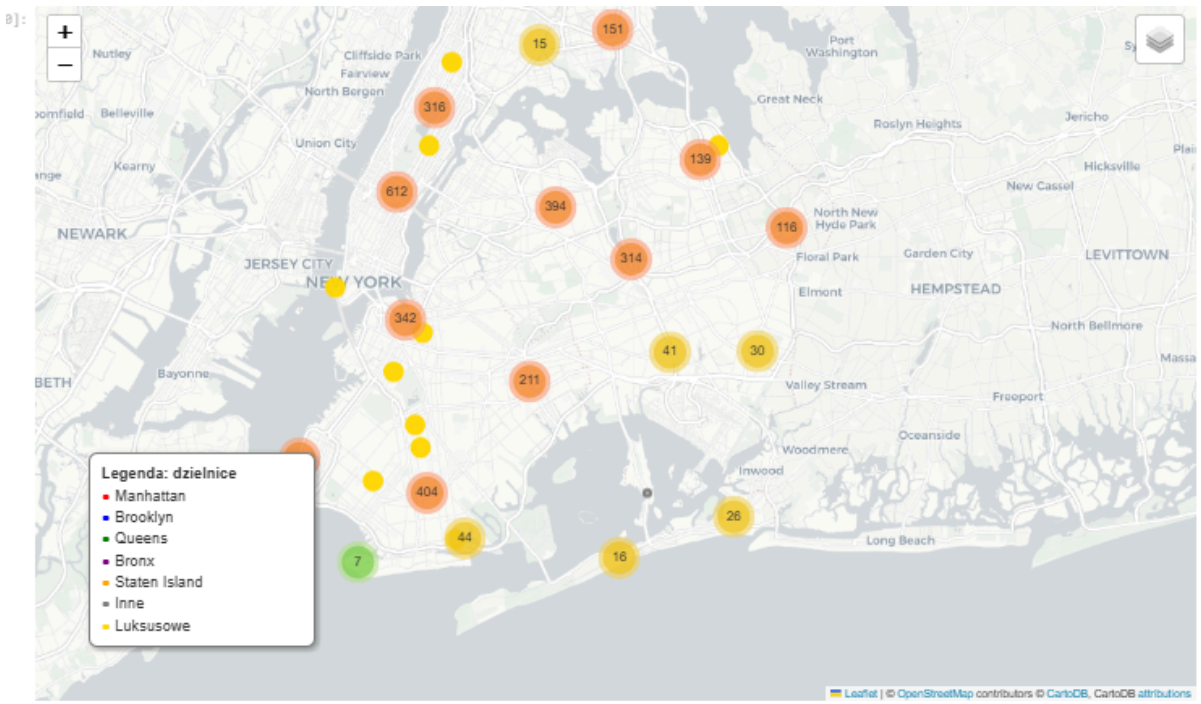
The division is based on actual New York City boroughs rather than artificial zones.

This is, of course, a simplification — accurate boundaries would require a shapefile and the use of tools like GeoPandas.

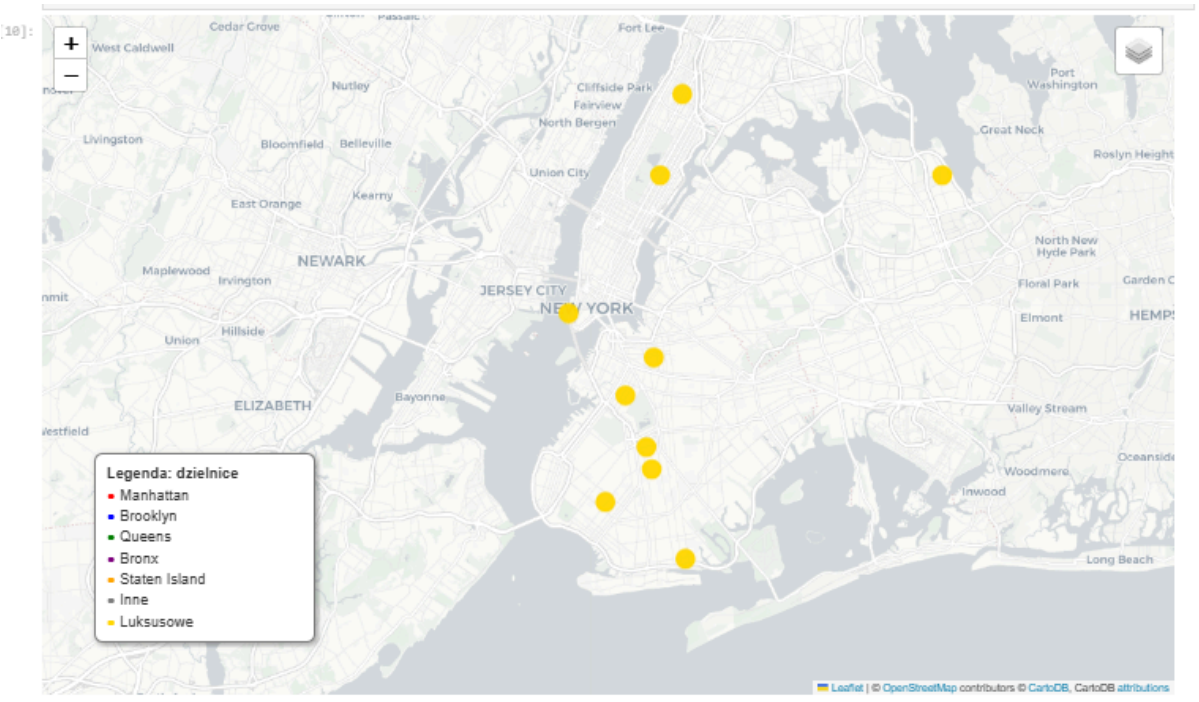
```
BOROUGH
Brooklyn         1020
Queens            831
Manhattan         800
Other             536
Staten Island     181
Bronx             130
Name: count, dtype: int64
```

More intuitive, aligned with the real estate market, and easier to interpret for both the user and the model.

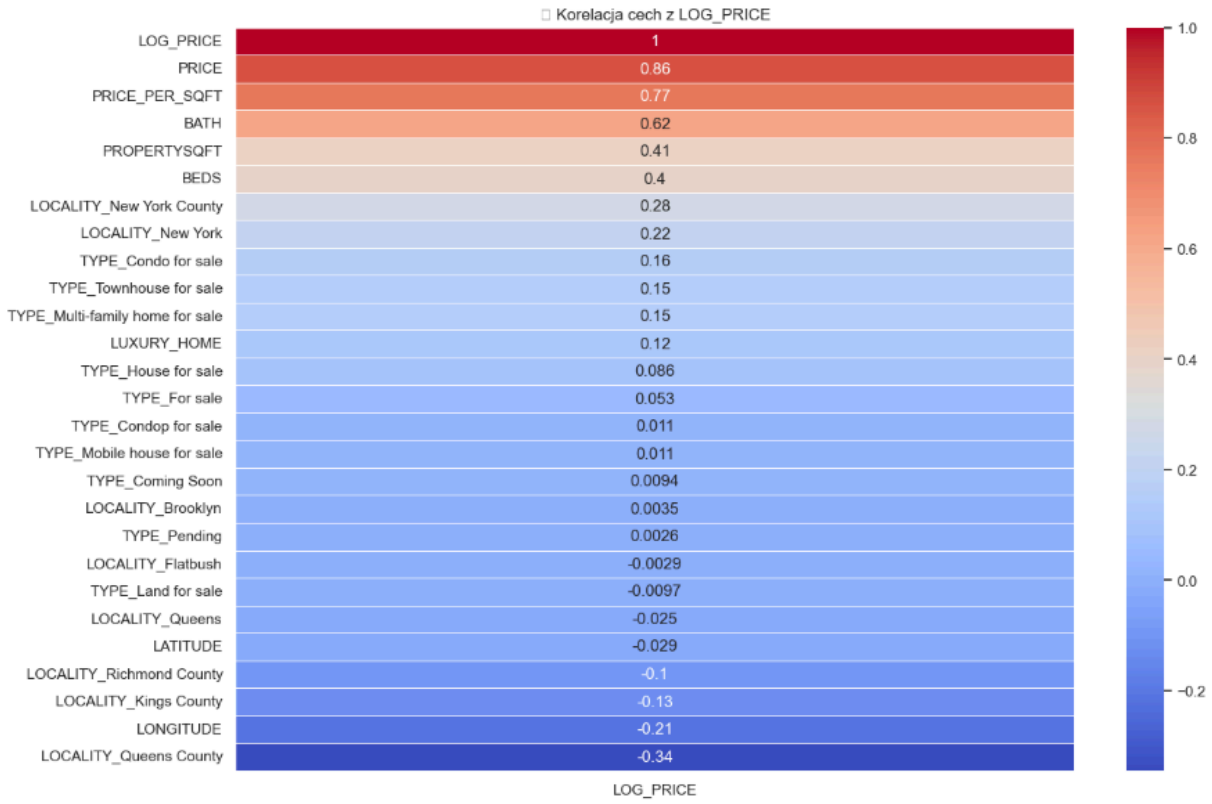
Map showing properties by borough (BOROUGH)



Luxury homes layer



Heatmap of feature correlations with PRICE



Conclusions:
The strongest impact on property value comes from the price itself and the price per square foot (PRICE_PER_SQFT), which aligns with the economic fundamentals of the real estate market.
Comfort features such as the number of bathrooms (BATH) and property size (PROPERTYSQFT) also have a significant, though slightly lower, influence. Location (e.g., Queens County) can affect price positively or negatively, depending on the specific region.

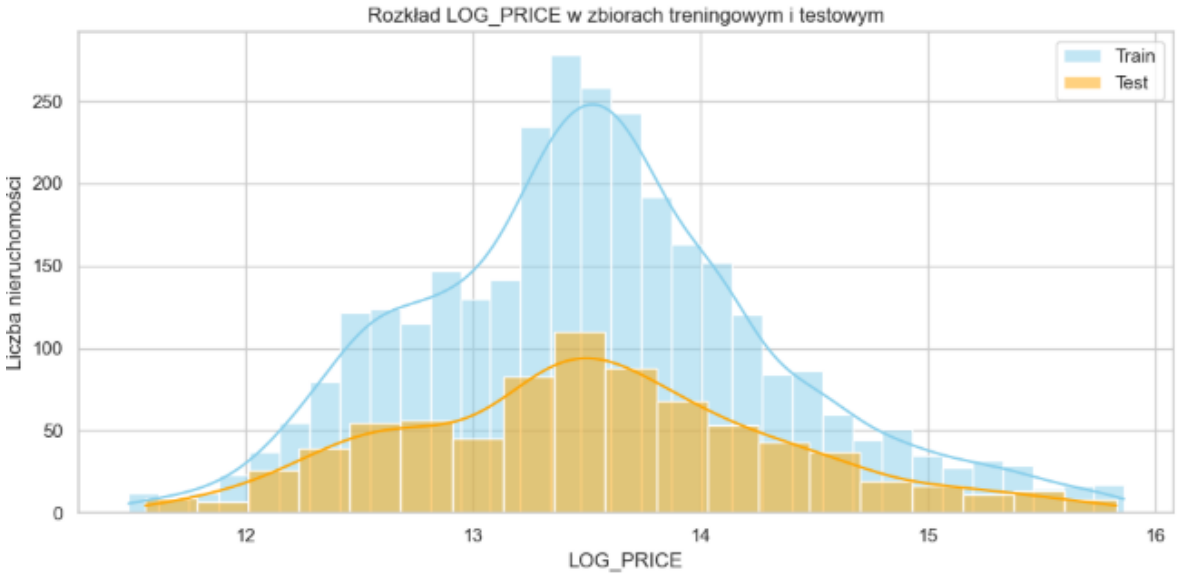
Preparing training and testing sets

This involves splitting the data into:

- X – features (input variables)
- y – target variable (LOG_PRICE)

Then performing a train/test split, e.g., 80% for training and 20% for testing.

Distribution of the target variable in train/test sets



The training and testing sets have very similar distributions, indicating that the split was performed correctly.

There are no extreme differences (e.g., the test set containing only cheap or only expensive properties).

This is a strong indication of the representativeness and reliability of the test data.

In the dataset X, there are three text columns: 'BOROUGH', 'LOCATION_CATEGORY', and 'TYPE', where One-Hot Encoding was applied, since linear regression only accepts numerical input data.

Mean (baseline).

A model based solely on the mean of the target variable shows a high average prediction error an expected result, as the mean does not take any feature information into account.

```
Baseline (średnia) RMSE: 0.8311
Baseline R²: -0.0000
```

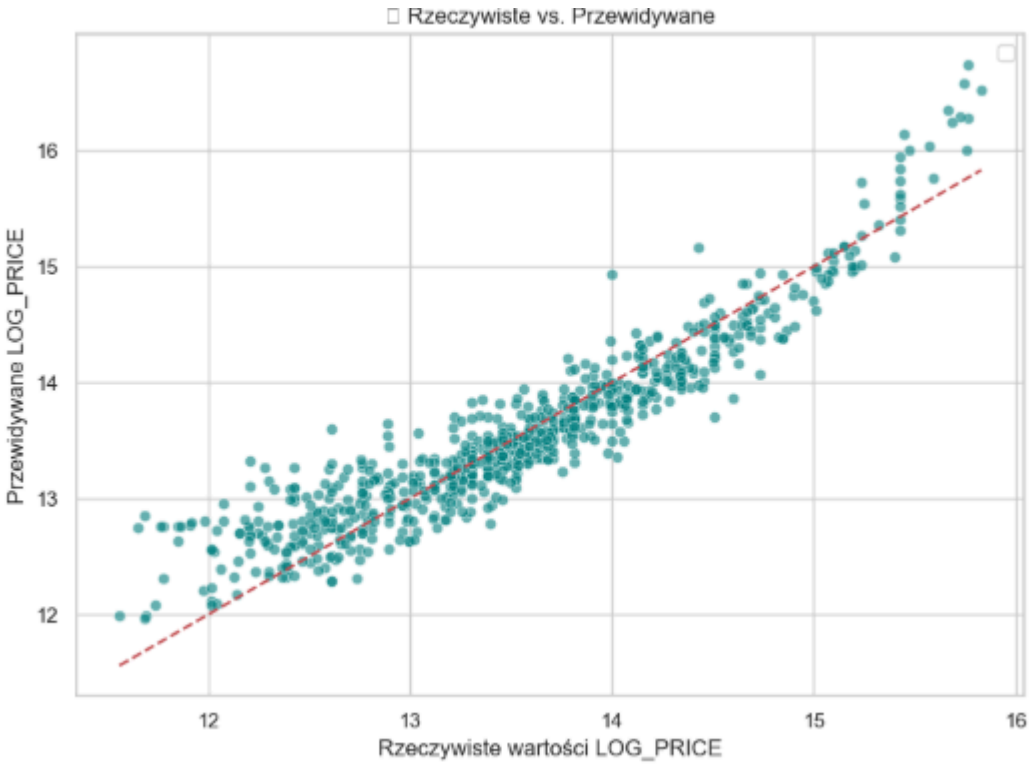
Linear Regression.

```
Regresja liniowa RMSE: 0.2990
Regresja liniowa R²: 0.8705
```

Results: Mean (Baseline) vs. Linear Regression

A reduction of RMSE by over 60% compared to the baseline indicates that linear regression predicts the target variable much more accurately and significantly reduces prediction errors.

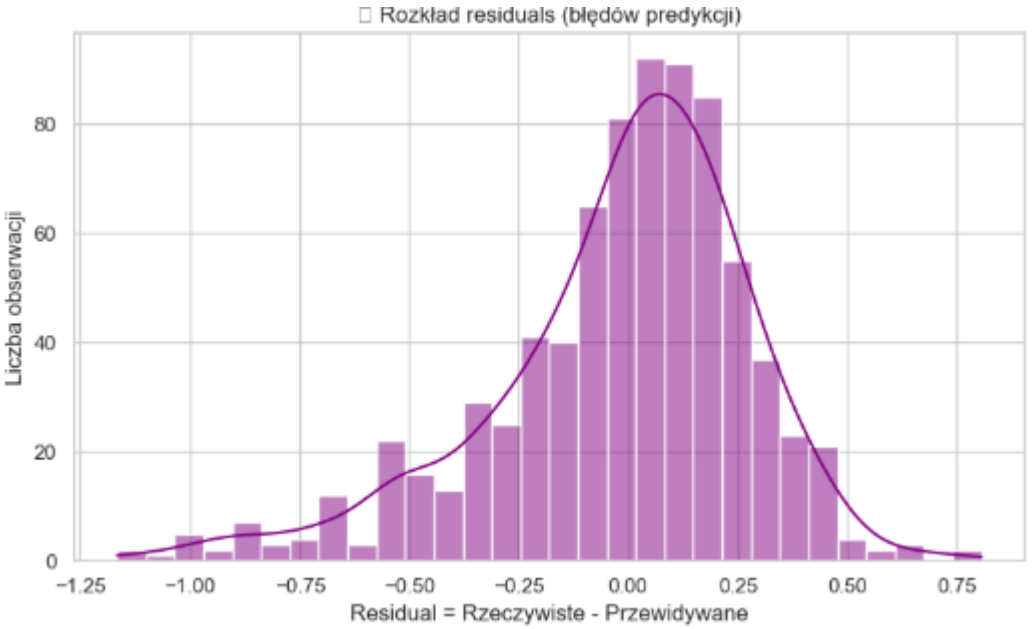
Actual vs. Predicted



Analysis

The data aligns well with the ideal line, confirming the model's effectiveness in predicting `LOG_PRICE`. Minor deviations indicate high precision, while a few larger errors point to more challenging cases. These results validate the importance of the selected features in the predictive process.

Residuals Plot (Prediction Errors)



Conclusions

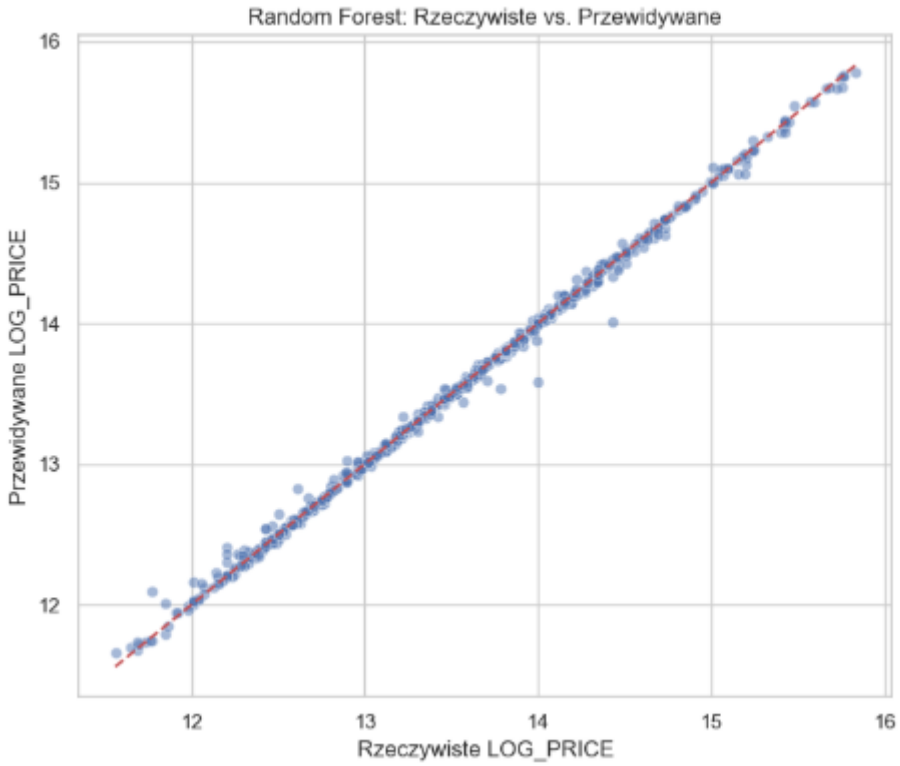
The distribution of residuals is symmetric and centered around zero, indicating good performance of the linear regression model.

- Deviations from zero may point to properties with atypical features that are harder to predict accurately.
- The lack of skewness in the residuals suggests that the model does not systematically overestimate or underestimate the target variable.
- This behavior confirms that the model assumptions are well met and the predictions are generally unbiased and reliable.

Random Forest Regressor

```
Random Forest RMSE: 0.0403
Random Forest R²: 0.9976
```

Actual vs. Predicted, i.e., how close the model gets to the true values.

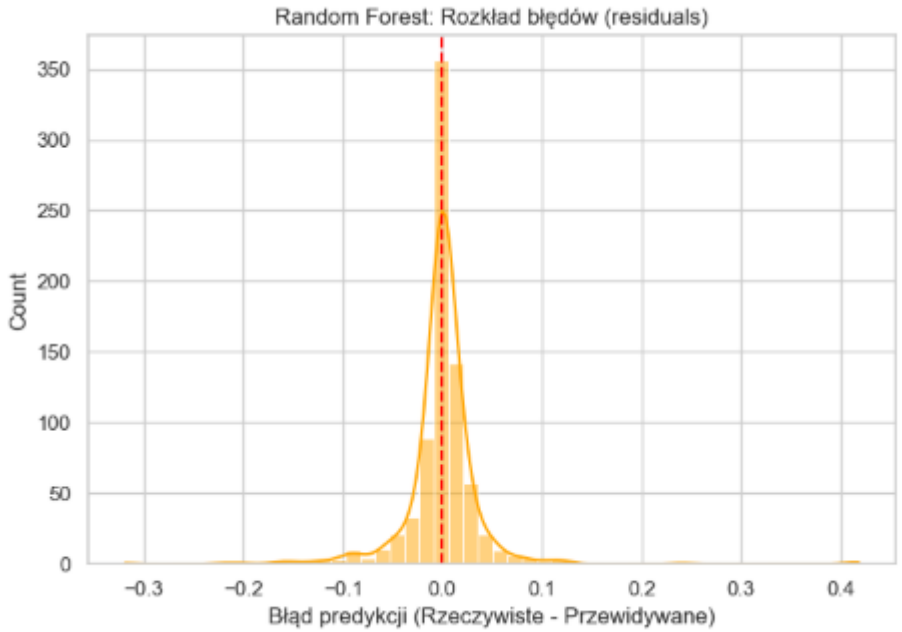


Conclusions

Points along the diagonal line: Most points on the plot are very close to the red line (perfect fit: actual = predicted), which indicates high precision of the Random Forest model in predicting LOG_PRICE.

Minimal deviations: A few observations may indicate properties with unusual features or outliers, where the model had difficulty achieving a perfect fit.

Residuals Plot (Prediction Errors)



Analysis

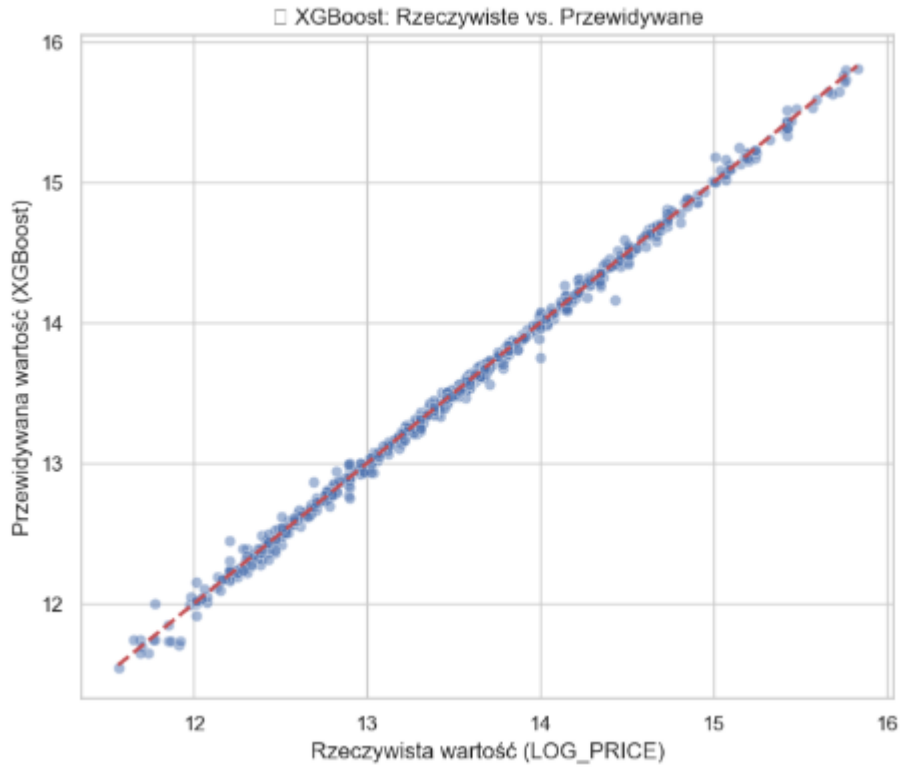
The histogram shows a distinct peak around the value 0 (prediction errors are concentrated near zero). This indicates that most prediction errors are very small. The distribution of errors is symmetrical, suggesting that the model does not exhibit significant tendencies to systematically overestimate or underestimate values. Prediction errors are tightly clustered around zero.

XGBoost

```
XGBoost RMSE: 0.0423
XGBoost R²: 0.9974
```

The RMSE value indicates that XGBoost predicts the target variable (LOG_PRICE) very well, with an error only slightly higher than Random Forest (0.0403). The model explains 99.74% of the variance in the data, which reflects an almost perfect capture of patterns and relationships within the dataset.

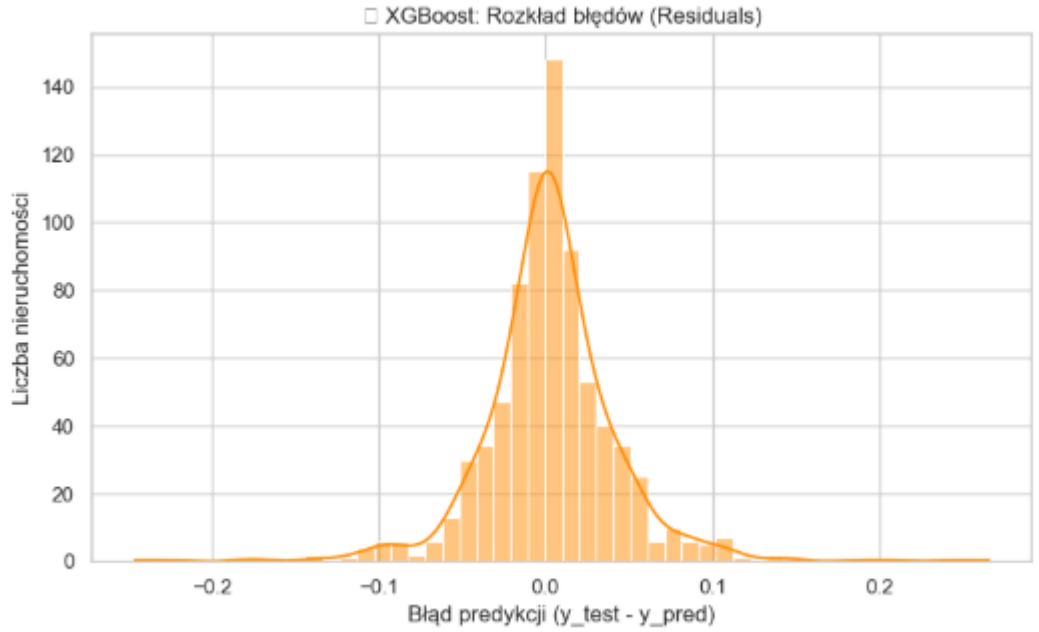
Actual vs. Predicted



Analysis

The XGBoost model accurately reflects the actual LOG_PRICE values – most data points lie close to the ideal fit line. Minor deviations are limited to a few atypical cases.

Residuals Plot (Prediction Errors)



Analysis

The histogram shows a strong concentration of errors near 0. Most prediction errors are minimal, indicating high precision of the XGBoost model. The errors are evenly distributed around 0, which suggests no systematic underestimation or overestimation of results. The vast majority of errors fall within the range of -0.2 to 0.2, with only a few observations showing larger deviations.

Hyperparameter Tuning. Grid Search, Optimization.

The goal is to find the best parameter settings that improve the model's accuracy without overfitting.

RandomForestRegressor model settings

'n_estimators': [100, 200]	Number of trees in the forest
'max_depth': [None, 10, 20]	Maximum depth of the tree
'min_samples_split': [2, 5]	Minimum number of samples required to split a node
'min_samples_leaf': [1, 2]	Minimum number of samples required to be at a leaf node

Tested 24 parameter combinations (2 × 3 × 2 × 2).

For each combination, cross-validation was performed (cv=5).

The model selects the best set of hyperparameters based on the lowest RMSE.

GridSearchCV performs 5 validations for each of the 24 combinations = 120 training runs.

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
   Najlepsze parametry: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Random Forest (po optymalizacji) RMSE: 0.0398
Random Forest (po optymalizacji) R²: 0.9977
```

Conclusions

A higher number of trees proved more effective. No depth limitation (None) performed better than restrictions to 10 or 20, allowing the trees to capture complex patterns. The minimum number of samples to split a node is 2, enabling the model to make more detailed splits. The minimum number of samples per leaf is 1, which increases the model's precision.

Model Performance

RMSE = 0.0398. An even lower prediction error than before optimization (previously 0.0403). The model has become more precise. **R^2 = 0.9977.** Slightly better than the previous R^2 (0.9976), confirming the model's excellent fit to the data. The optimized model is ready for use in practical applications where high prediction accuracy is required.

XGBoost Model Settings

'n_estimators': [100, 200]	Number of trees in the ensemble.
'max_depth': [3, 6, 10]	Maximum depth of each tree.
'learning_rate': [0.01, 0.1]	Learning rate (shrinkage factor) used to prevent overfitting.
'subsample': [0.8, 1.0]	Fraction of the training data used for each boosting round.

5-fold cross-validation.

The data is divided into 5 parts. The model is trained 5 times, each time on 4 parts and tested on the remaining one. The evaluation is more reliable because it is based on multiple data splits. It also enables full utilization of computing power and accelerates the training process.

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
 Najlepsze parametry: {'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 200, 'subsample': 0.8}
 XGBoost (po optymalizacji) RMSE: 0.0335
 XGBoost (po optymalizacji) R²: 0.9984
```

Grid Search performed 5-fold cross-validation for all 24 parameter combinations, resulting in a total of 120 model training runs.

Conclusions

- RMSE (0.0335): The very low prediction error after optimization indicates that XGBoost predicts the value of `LOG_PRICE` with high precision.
- R^2 (0.9984): An R^2 score close to 1 means that the model explains 99.84% of the variance in the data, reflecting an almost perfect fit.
- The best parameters ensure a balance between model complexity and its ability to generalize.
- Optimized XGBoost achieves the best performance among all models tested.

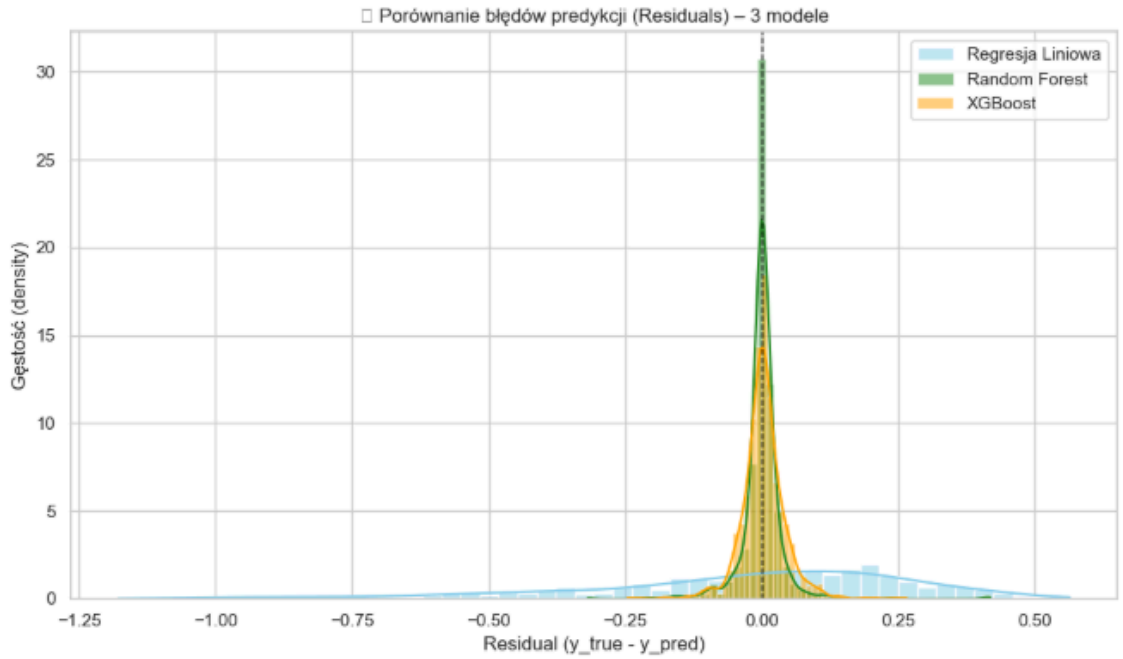
Comparison of Results

Model	RMSE	R^2	Remarks
Mean (baseline)	0.8311	-0.0000	No modeling, just the mean
Linear Regression	0.2990	0.8705	Simple model, but performed well
Random Forest Regressor	0.0403	0.9976	Very precise model
Random Forest (tuned)	0.0398	0.9977	Minimal improvement
XGBoost	0.0423	0.9974	Very good result
XGBoost (tuned)	0.0335	0.9984	Best prediction quality

Conclusions

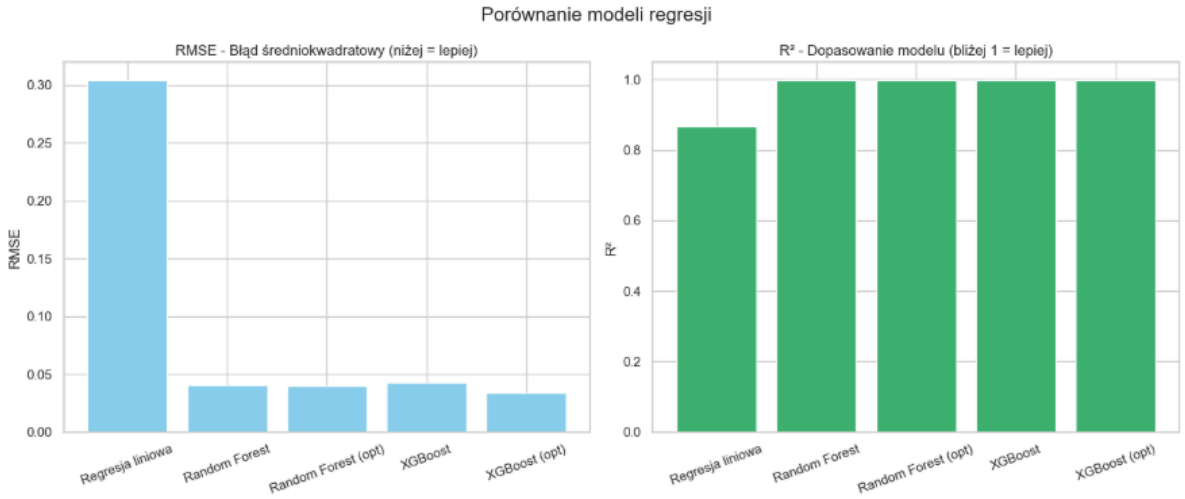
The baseline model (mean) does not provide any predictive value. Although simple, linear regression delivers solid results. Random Forest and XGBoost achieve very high precision, especially after hyperparameter tuning. The best performance was achieved by the optimized XGBoost model, making it the most effective model in this comparison.

Histogram of Prediction Errors (Residuals) – Model Comparison



Analysis

Histograms of residuals clearly show differences in prediction quality between models. Linear regression exhibits a wide and dispersed error distribution, indicating higher variability and lower precision. In contrast, tree-based models — Random Forest and XGBoost — display a narrow, tall peak around zero, reflecting high prediction accuracy. XGBoost has the narrowest distribution, suggesting slightly better performance than Random Forest, although the differences are minimal.



API

Allows sending new data and receiving the predicted property price — in this case, the logarithm of the price (LOG_PRICE), or ultimately the rescaled (actual) price.

Local API with XGBoost Model

1. The XGBoost model was trained and saved.
2. A list of required columns (features) was saved.
3. A `app.py` file with a Flask server was created.

– POST Endpoint
– `/predict`
– Accepts data in JSON format
– Returns the predicted LOG_PRICE as JSON

4. The API was successfully launched locally.
5. A sample test request was sent from Jupyter Notebook and processed correctly.

```
Używane cechy przez model:
- BEDS
- BATH
- PROPERTYSQFT
- LATITUDE
- LONGITUDE
- LOCALITY_Brooklyn
- LOCALITY_Flatbush
- LOCALITY_Kings County
- LOCALITY_New York
- LOCALITY_New York County
- LOCALITY_Queens
- LOCALITY_Queens County
- LOCALITY_Richmond County
- TYPE_Coming Soon
- TYPE_Condo for sale
- TYPE_Condop for sale
- TYPE_For sale
- TYPE_House for sale
- TYPE_Land for sale
- TYPE_Mobile house for sale
- TYPE_Multi-family home for sale
- TYPE_Pending
- TYPE_Townhouse for sale
- LUXURY_HOME
- PRICE_PER_SQFT
- BOROUGH_Brooklyn
- BOROUGH_Manhattan
- BOROUGH_Other
- BOROUGH_Queens
- BOROUGH_Staten Island
- LOCATION_CATEGORY_NorthEast
- LOCATION_CATEGORY_Other
- LOCATION_CATEGORY_SouthWest
```

Test Input data

Feature	Value	Description
Beds	3	Number of bedrooms in

		the property
Bath	2	Number of bathrooms
Propertysoft	100	Property area in square meters ≈ 1076 square feet
Lattude	40.7306	Geographical latitude (example: Manhattan)
Longitude	-73.9352	Geographical longitude
Price per soft	1100	Price per square foot in USD
Luxury home	1	Whether the property is considered luxury (1 = yes, 0 = no)

Categorical features

Cecha	Wartość	Znaczenie
Borough Manhattan	1	The property is located in the **Manhattan** borough
Type Condo for sale	1	The property type is **Condo for sale**
Location Category north east	1	Location classified as **NorthEast** based on coordinates
Locality New York Country	1	More precise location – **New York County**, which corresponds to **Manhattan**

All other categories have a value of 0, meaning they do not apply to this specific property.

Predicted Market Price

```
Logarytmiczna prognoza ceny (LOG_PRICE): 12.9593
Prognozowana cena nieruchomości: $424,761.80
```

HTML Form

Formularz predykcji ceny nieruchomości

Dane numeryczne

BEDS:
2

BATH:
1

PROPERTYSQFT:
65

LATITUDE:
40.7306

LONGITUDE:
-73.9352

PRICE_PER_SQFT:
6500

LUXURY_HOME (0 lub 1):
0

LOCALITY

- [ ] Brooklyn
- [ ] Flatbush
- [x] Kings County
- [ ] New York
- [ ] New York County
- [ ] Queens
- [ ] Queens County
- [ ] Richmond County

TYPE

- [x] Coming Soon
- [ ] Condo
- [ ] Condop
- [ ] For sale
- [ ] House
- [ ] Land
- [ ] Mobile house
- [ ] Multi-family
- [ ] Pending
- [ ] Townhouse

BOROUGH

- [ ] Brooklyn
- [x] Manhattan
- [ ] Other
- [ ] Queens
- [ ] Staten Island

LOCATION CATEGORY

- [x] NorthEast
- [ ] Other
- [ ] SouthWest

Przewiduj cenę

Logarytmiczna prognoza ceny (LOG_PRICE): 13.6399

Prognozowana cena nieruchomości: $838,907.19

← Wróć do formularza

Final Conclusion

The XGBoost model proved to be the most accurate (RMSE = 0.0335, R^2 = 0.9984), outperforming both Linear Regression and Random Forest. Moreover, the XGBoost model optimized via GridSearchCV achieved excellent fit on the test data while maintaining strong generalization capabilities.

The implementation of the API allows for rapid prediction of real estate values based on real-world input data.

Square Footage vs Logarithmic Price

Analysis

Historical data (blue dots):
The scatter plot illustrates the relationship between property size (PROPERTYSQFT) and logarithmic price (LOG_PRICE). A clear upward trend is visible larger properties generally correspond to higher log prices.
There is significant variability in price for smaller properties, indicating that other factors (such as location or property type) also strongly influence price.

API prediction result (red dot):
The red dot represents the predicted LOG_PRICE for a new data input submitted via the API. It

appears around the point (65, 12.5) corresponding to a property of 65 m² and a log price close to 12.5.

The predicted value aligns well with the trend in historical data, demonstrating that the model effectively captures the correlation between size and price.
The predicted price reflects the combination of input features such as non-luxury status and Manhattan location contributing to the final valuation.

The chart not only illustrates the relationship between size and price, but also highlights the effectiveness of the API in generating accurate predictions in line with real-world data.

.