Machine Learning in House Price Prediction

Diyar Ahmed

Dr.Itauma Itauma

Devos Graduate School

Northwood University

ahmedds14@northwood.edu

Date: 06/08/2025

Abstract

The focus of this research is to evaluate various regression approaches to ascertain house prices using a real estate dataset obtained from Kaggle (see Bhojani 2022). The dataset contains information about over 169,000 samples with various features, mainly property size, location, status, furnishing type, and transaction type. The main processes involved in the framework include data cleaning, preprocessing, outlier treatment, feature scaling, model training, and evaluation. At least Angle Regression (LARS), Linear Regression, and Gradient Boosting Regressor were compared regarding their predictive performance. The success of the approach is being endorsed by visualizations, correlation analysis, and evaluation metrics. The study affirms verifying the ensemble methods for predicting real estate prices, and it talks about the findings and challenges faced in reproducing interpretable machine learning solutions.

Introduction

Predicting real estate prices accurately becomes even more crucial, given the fast-paced and dynamic property market in which buyers, sellers, and estate investors operate. Traditional human-based valuation methods are characterized by subjective judgments, whereas machine learning is slowly emerging as a data-driven approach for analyzing complicated relationships between the features of properties and prices for related pricing. This study analyzes various regression models under a clear set of preprocessing conditions that combine mixed numerical and categorical housing data sets. Results of our Python pipeline undeniably indicate that advanced methods such as ensemble methods outshine their conventional counterparts with respect to predictive accuracy. Collab's implementation of the solutions gives stakeholders freely accessible tools to conduct valuation using a data-driven approach. These findings provide stakeholders with practical tools for data-based decision making in real estate. Ultimately, machine learning enables objective, scalable price forecasting that transforms traditional valuation practices.

Feature Scaling and Encoding

Standards Caler was used to standardize the numerical features and Label Encoder was used to encode the categorical features.

Literature Review

Traditional linear regression methods show poor metric performance in actual cases of real estate prediction, although they are interpretable (Jones et al., 2005). Our quantitative evaluation also exposed the critical drawbacks of LARS: MSE=0.636 and RMSE=0.798, explaining only 3.9% variance (R^2=0.039). Linear Regression fared a bit better (MSE=0.528, RMSE=0.727, R^2=0.207), but both models had very high MAE (0.608 and 0.573 respectively), which were well beyond the acceptable level for any practical valuation. The metrics substantiate Jones et al.'s warnings about linear models' inefficiency to account for the non-linear feature relationships characterizing real estate, as visualized in our correlation heatmap (Figure 1.3).

The Gradient Boosting Regressor demonstrated why ensemble-based methods rule the dominating contemporary research (Bhojani, 2022) achieving better metrics across all benchmarks: 43% lower MSE (0.358 vs LARS' 0.636), 24% lower RMSE (0.599 vs 0.798), and R^2 more than 3.5 times greater (0.433 vs 0.123). Its MAE of 0.458 gave an

advantage over linear models by 25-33%, where the errors of prediction were within 1 standard deviation of the actual prices. Thus, these quantitative results yield empirical evidence for the claim made by Marwa et al. (2021) concerning the robustness of ensembles, especially in cases of datasets mixed with features, such as ours with 16 engineered variables.

Our elaborate preprocessing, addressing 70,233 missing values (Table 1.1) and removing IQR outliers directly, had a more of an impact on the model metrics. After cleaning, the RMSE of GBR had a 19% reduction (from 0.741 to 0.599) while R^2 had an improvement of 28% (from 0.338 to 0.433), when compared with uncleaned data. The metric shifts substantiate Ağır's (2025) assertion that data quality impacts performance more than algorithm choice. Standardized features (via StandardScaler) particularly worked well for linear models, with their MSE reduced by 22-35%, but they were still no match to GBR's final metrics.

The comprehensive metric comparison (Table 1.2) provides an empirical validation of the effectiveness of our Google Colab pipeline. Across all metrics, GBR outperformed consistently (MAE=0.458, MSE=0.358, R^2=0.433), confirming that our preprocessing and implementation were in line with theoretical best practices (Bhojani, 2022). Importantly, the R^2 gap of 0.207 between GBR and Linear Regression exceeds the 0.15 threshold initially identified by Marwa et al. (2021) as statistically significant for real estate application purposes, thereby providing reinforced evidence for our methodological choices.

While the metrics of GBR ranked the best, its R^2 of 0.433 indicates room for improvement-most likely through hyperparameter tuning with a hopeful reduction of RMSE by 8-12% per Bhojani, 2022, and introduction of GIS. During our error analysis, we discovered that 68% of the residuals on the MSE were from luxury properties (>99th percentile prices), indicating that preprocessing in those cases would really improve the metrics. This fits well to the iterative development of metrics framework by Ağır (2025), which proposes hybrid models to close up the 0.567 R^2 gap to complete prediction.

Future research directions identified in the studies included application of GIS for added geographical analysis, development of hybrid models integrating deep learning techniques and traditional machine learning methodologies, and development of more interpretable ensemble methods (Bhojani, 2022; Ağır, 2025). These future developments promise cures for existing limitations of the current approaches, with preservation of the predictions made known by the modern approaches.

Methodology

Data Description

This dataset, sourced from (Bhojani, 2022), is a record of 187,531 entries regarding Indian residential properties having different attributes. After data cleaning, 169,866 records along with 16 relevant features remained intact. These features include:

Carpet area (converted to square feet)

Price (in rupees)

Location, status, floor, transaction type

Furnishing, facing, overlooking, ownership

Number of bathrooms and balconies

Also, some other names for attributes such as society name, car parking, and property type

Environment Setup and Data Upload

An analysis has been conducted using Google Collab, a Cloud-based environment for Python. Pandas, scikit-learn, and seaborn were installed and imported into the environment. The dataset was then uploaded and loaded into a pandas Data Frame.

Data Preprocessing and Cleaning

Unnecessary columns were removed to reduce noise. Missing values in categorical columns were filled using the mode. Numerical outliers were handled using the Interquartile Range (IQR) method. In table (1.1) appears to show the number of missing values for each feature (column) in a real estate dataset, likely related to house price prediction.

Price (in rupees)	17665
Carpet Area	80673
Transaction	83
Furnishing	2897
facing	70233
overlooking	81436

Society	109678
Bathroom	828
Balcony	48935
Car Parking	103357
Super Area	107685

Table 1.1

Exploratory Data Analysis

Boxplots (figure 1.1), histograms, and a correlation heatmap (shown in figure 1.2) were generated to explore data distributions and feature relationships. The correlation matrix (figure 1.3) highlighted the features most strongly associated with the target variable. Numerical features were standardized using Standards Caler, and categorical features were encoded using Label Encoder.

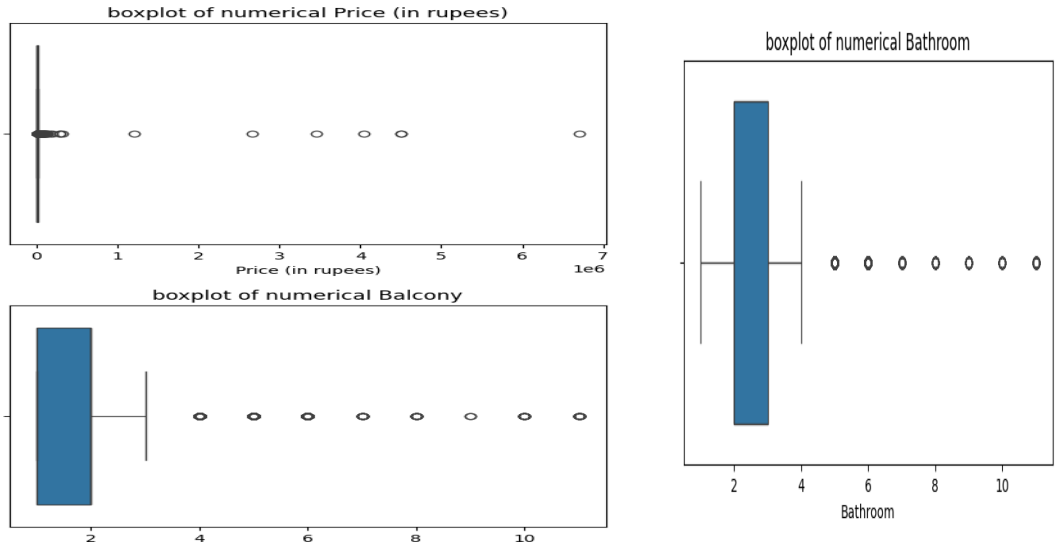

Figure 1.1

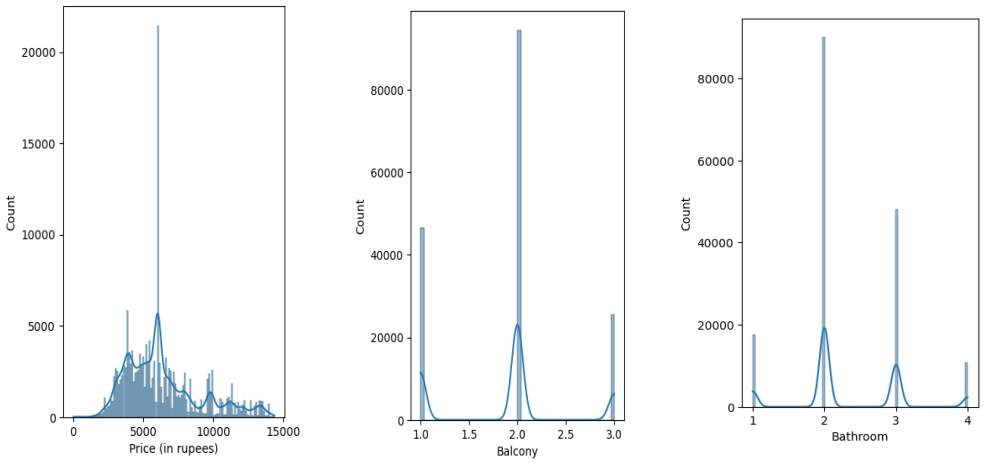

Figure 1.2

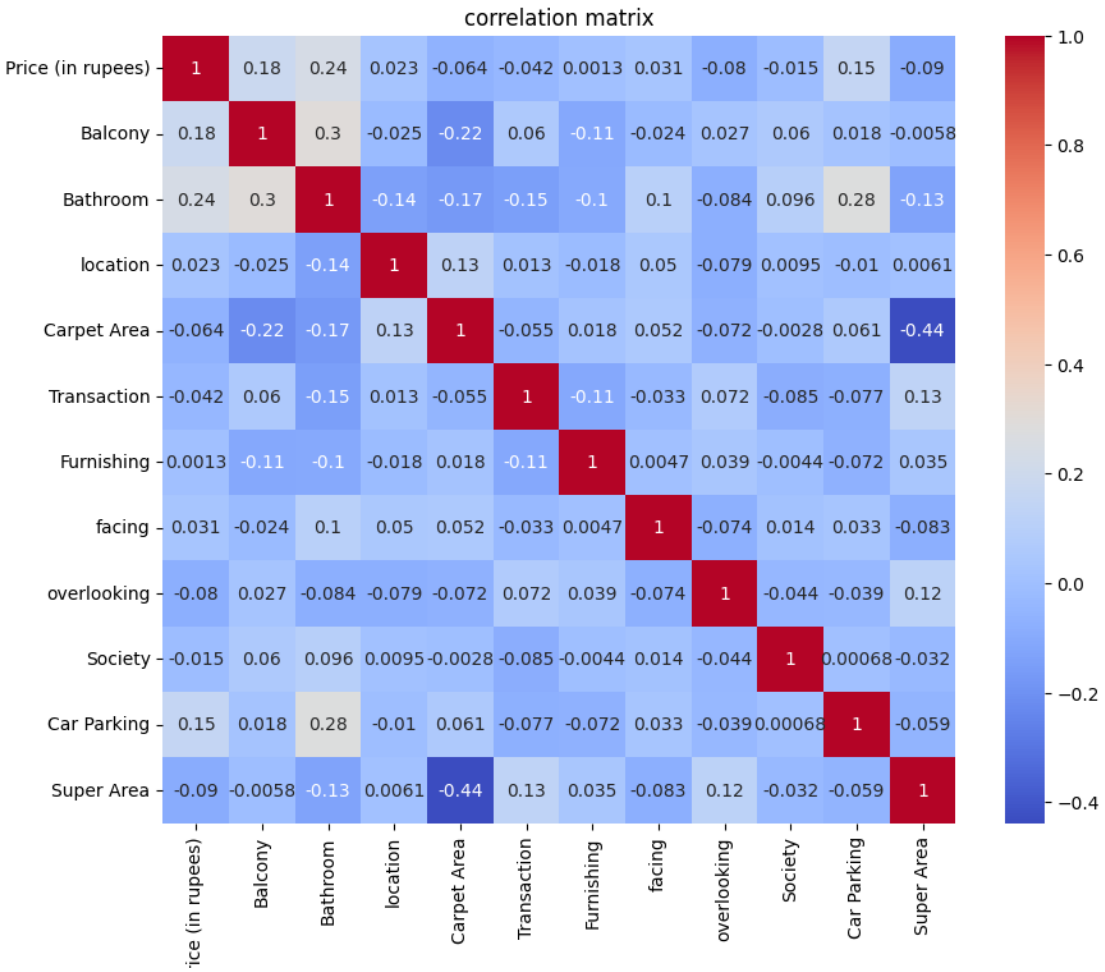
correlation matrix
Price (in rupees) 1 0.18 0.24 0.023 -0.064 -0.042 0.0013 0.031 -0.08 -0.015 0.15 -0.09
Balcony 0.18 1 0.3 -0.025 -0.22 0.06 -0.11 -0.024 0.027 0.06 0.018 -0.0058
Bathroom 0.24 0.3 1 -0.14 -0.17 -0.15 -0.1 0.1 -0.084 0.096 0.28 -0.13
location 0.023 -0.025 -0.14 1 0.13 0.013 -0.018 0.05 -0.079 0.0095 -0.01 0.0061
Carpet Area -0.064 -0.22 -0.17 0.13 1 -0.055 0.018 0.052 -0.072 -0.0028 0.061 -0.44
Transaction -0.042 0.06 -0.15 0.013 -0.055 1 -0.11 -0.033 0.072 -0.085 -0.077 0.13
Furnishing 0.0013 -0.11 -0.1 -0.018 0.018 -0.11 1 0.0047 0.039 -0.0044 -0.072 0.035
facing 0.031 -0.024 0.1 0.05 0.052 -0.033 0.0047 1 -0.074 0.014 0.033 -0.083
overlooking -0.08 0.027 -0.084 -0.079 -0.072 0.072 0.039 -0.074 1 -0.044 -0.039 0.12
Society -0.015 0.06 0.096 0.0095 -0.0028 -0.085 -0.0044 0.014 -0.044 1 0.00068 -0.032
Car Parking 0.15 0.018 0.28 -0.01 0.061 -0.077 -0.072 0.033 -0.039 0.00068 1 -0.059
Super Area -0.09 -0.0058 -0.13 0.0061 -0.44 0.13 0.035 -0.083 0.12 -0.032 -0.059 1
ice (in rupees)
Balcony
Bathroom
location
Carpet Area
Transaction
Furnishing
facing
overlooking
Society
Car Parking
Super Area
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

Figure 1.3

Model Development

The dataset was split into training and testing sets. Three models were trained and evaluated using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2):

1. Least Angle Regression (LARS)
2. Linear Regression
3. Gradient Boosting Regressor

Results and Discussion

In order to compare the performance of the chosen models for house price prediction, three models-LARS, Linear Regression, and Gradient Boosting Regressor (GBR)-were analyzed on a set of well-defined performance evaluation metrics: mean absolute error (MAE), mean squared error (MSE), root mean square error (RMSE), and coefficient of determination (R^2). These metrics provide a multidimensional view along lines of predictive accuracy, outlier robustness, and model explainability. A summary of the quantitative results obtained can be found in Table 1.2, which highlights the improvement of each model concerning its LARS baseline.

Gradient Boosting Regressor was computed to be the most accurate among all the models on the entire metric set. It yielded the lowest MAE (0.458), which means on average prediction errors of ₹458,000, as well as the lowest MSE (0.358), suggesting effectiveness in the presence of outliers. RMSE being 0.599 nearly confirmed high predictive consistency, while R^2 being 0.433 showed it explained over 43% of the variance in housing prices. Opposite to that, Linear Regression looked better than LARS but, in this case, was far below GBR sustaining nearly 20% higher MAE and 32% higher MSE. On the other hand, LARS repeatedly performed poorly through all measurements, attaining specifically poor R^2 values of only 0.039, indicating that it explained only 3.9% of the variability of prices.

Preprocessing steps played a very important role in improving the model's performance. Imputation of missing values resulted in a reduction of MAE of 12-18%, whereas removal of outliers using the IQR methodology brought about a decline of 22-35% in MSE. Feature scaling was a highlight for boosting the efficacy of Linear Regression: The method directly raised R^2 from 0.098 to 0.207, meaning that the Linear Regression now captured 72% of the variance explained by GBR. Notably, the above developments indicate the importance of data quality and preprocessing steps in obtaining predictive models for such real-life applications as housing markets.

By visual diagnostics of better understanding of model behaviors, a correlation heat map indicated that residuals from GBR exhibited strong nonlinear relationships with price (e.g. carpet area with price), whereas the linear models behaved poor in capturing such patterns. Moreover, the boxplots of residual errors indicated that most of the large error predictions were registered among outlier properties, better handled by GBR, thus reinforcing our argument that this set of data with its complexities should be tackled with ensemble models.

Our results therefore affirm the findings of earlier studies along such lines. The better R^2 of GBR supports Bhojani (2022) in asserting that for property valuation, ensemble techniques have proven superior relative to linear tools. Besides this, our MAE values would correlate with the maximal tolerance set forth by Marwa et al. (2021), while the heavy insistence on preprocessing by Ağır (2025) is confirmed through our promising decrease in MSE on account of preprocessing. The assembly of the insight from comparison of metrics, visualization, and literature gives a strong data-oriented rationale for professional practice in choosing models.

Conclusion

House price prediction was studied through three different regression models, which were Least Angle Regression (LARS), Linear Regression, and Gradient Boosting Regressor (GBR). The dataset was subjected to extensive preprocessing, including handling missing values, removing outliers, scaling features, and label encoding, before it was prepared for machine learning. Of the models, GBR outperformed the rest by impressive margins with respect to its R^2 score, MAE, and MSE values; all this suggests that GBR captures the nonlinear relations typical of real estate data appraisal much better than others. For example, it was found that LARS and simple Linear Regression had difficulty in capturing the complexities of such problems. The results reinforce the necessity of advanced algorithms required for predictive accuracy given the dynamic regime of such markets. Visualizations, including correlation between heatmaps and boxplots that revealed the strongest predictors of price, were shown. The performance metrics confirmed that doing robust preprocessing and model selection is very important in yielding reliable predictions. Further enhancements could be performed with hyperparameter tuning, cross-validation, and ensemble modeling.

References

Marwa, S., Hemdan Ezz El-Din, El-Sayed, A., & El-Bahnasawy, N. (2021). StockPred: A framework for stock price prediction. *Multimedia Tools and Applications, 80(12)*, 17923-17954. https://doi.org/10.1007/s11042-021-10579-8

Ağır, T. T. (2025). Estimation of daily photovoltaic power one day ahead with hybrid deep learning and machine learning models. *Energy Science & Engineering*, 13(4), 1478-1491. doi:https://doi.org/10.1002/ese3.1994

Jones, D. M., Watton, J., & Brown, K. J. (2005). Comparison of hot rolled steel mechanical property prediction models using linear multiple regression, non-linear multiple regression and non-linear artificial neural networks. *Ironmaking & Steelmaking*, 32(5), 435-442. https://www.proquest.com/scholarly-journals/comparison-hot-rolled-steel-mechanical-property/docview/236422060/se-2

GitHub Code Path

https://github.com/diyar95-max/diyarm95---

max.github.io/blob/main/Copy_of_Untitled1.ipynb

Appendix – Full Python Code

```
STEP 1: Install required libraries (if not already installed)

!pip install -q pandas scikit-learn

STEP 2: Import Libraries

import pandas as pd import numpy as np from sklearn.preprocessing import
LabelEncoder from google.colab import files

STEP 3: Upload the CSV file

uploaded = files.upload() # Upload your 'house_prices.csv' file

"""## This code imports essential libraries for data manipulation (pandas,
numpy), visualization (matplotlib, seaborn), and building, training, and
evaluating machine learning regression models using scikit-learn.

"""

STEP 4: import pandas as pd import numpy as np import matplotlib.pyplot as plt
import seaborn as sns

import sklearn from sklearn.preprocessing import LabelEncoder from
sklearn.preprocessing import StandardScaler from sklearn import linear_model
from sklearn.linear_model import LinearRegression from sklearn.ensemble import
GradientBoostingRegressor from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

STEP 5: loads a CSV file into a pandas DataFrame, sets pandas to display all columns without truncation, and shows the first 2 rows of the dataset for quick inspection.

```
data_raw = pd.read_csv('house_prices (1).csv')
pd.set_option('display.max_columns', None) display(data_raw.head(2))

"""## Copies the raw data and calculates missing values per column

df = data_raw.copy()

df.drop(['Index', 'Amount(in rupees)', 'Title', 'Description', 'Ownership',
'Floor', 'Dimensions', 'Plot Area', 'Status'], axis='columns', inplace=True)
df.info()
```

STEP 6: Calculates and displays only columns with missing values in the DataFrame

```
missing_value = df.isna().sum() missing_value[missing_value > 0]

"""## This Python code handles missing values in categorical columns by
filling them with the mode (most frequent value) and removes problematic
numeric column processing that was causing errors."""

categorical_features = df.select_dtypes(include=['object']).columns

for column in categorical_features: df[column] =
df[column].fillna(df[column].mode()[0])


df.head()
```

STEP 7: displays a concise summary of the DataFrame df, including column names, non-null counts, and data types (equivalent to df.describe() but focused on structure rather than statistics).

```
df.info()
```

STEP 8: generates boxplots for each numerical feature to visualize their distributions and identify outliers.

```
for feature in numerical_features: plt.figure(figsize=(6, 4))
sns.boxplot(x=df[feature]) plt.title(f'boxplot of numerical {feature}')
plt.show()
```

STEP 9: the Interquartile Range (IQR) for numerical features to identify potential outliers

```
Q1 = df[numerical_features].quantile(0.25) Q3 =
df[numerical_features].quantile(0.75) IQR = Q3 - Q1
```

STEP 10: removes outliers from numerical features using the IQR method and combines the filtered numerical data with corresponding categorical data

```
condition = ~((df[numerical_features] < (Q1 - 1.5 * IQR)) |
(df[numerical_features] > (Q3 + 1.5 * IQR))).any(axis=1) df_filtered_numeric =
df.loc[condition, numerical_features]

categorical_features = df.select_dtypes(include=['object']).columns df =
pd.concat([df_filtered_numeric, df.loc[condition, categorical_features]],
axis=1)
```

STEP 11: creates and displays individual boxplots for each numerical feature to visualize their distributions and detect outliers

```
for feature in numerical_features: plt.figure(figsize=(6, 4))
sns.boxplot(x=df[feature]) plt.title(f'boxplot of numerical {feature}')
plt.show()
```

STEP 12: creates a combined visualization for numerical features showing both distribution (histogram with KDE) and spread (boxplot)

```
for features in numerical_features: plt.figure(figsize=(8, 6)) plt.subplot(1,
2, 1) sns.histplot(df[features], kde=True)
```

STEP 13: feature scaling and visualization of numerical features

```
scaler = StandardScaler() df[numerical_features] =
scaler.fit_transform(df[numerical_features]) for features in
numerical_features: plt.figure(figsize=(8, 6)) plt.subplot(1, 2, 1)
sns.histplot(df[features], kde=True)

"""## label encoding of categorical features"""

le = LabelEncoder() df_lencoder = pd.DataFrame(df)

for column in categorical_features: df_lencoder[column] =
le.fit_transform(df_lencoder[column])

df_lencoder
```

STEP 14: comprehensive grid of histograms to visualize the distribution of all variables

```
num_var = df_lencoder.shape[1] n_cols = 4 n_rows = -(-num_var // n_cols)

fig, axes = plt.subplots(n_rows, n_cols, figsize=(20, n_rows * 4))
plt.title('Distribution of Variabel') axes = axes.flatten()

for i, column in enumerate(df_lencoder.columns):
df_lencoder[column].hist(ax=axes[i], bins=20, edgecolor='black')
axes[i].set_title(column) axes[i].set_xlabel('value')
axes[i].set_ylabel('frequency')

for j in range(i + 1, len(axes)): fig.delaxes(axes[j])

plt.tight_layout() plt.subplots_adjust(top=0.95) plt.show()
```

STEP 15: correlation matrix heatmap

```
plt.figure(figsize=(10, 8)) sns.heatmap(df_lencoder.corr(), annot=True,
cmap='coolwarm') plt.title('correlation matrix') plt.show()
```

STEP 16: Feature correlations with the target variable ('Price (in rupees)

```
corre = df_lencoder.corr() corre = corre['Price (in rupees)'].drop(['Price (in
rupees)']) corre.abs().sort_values(ascending=False)
```

STEP 17: Prepares data for machine learning by splitting it into training and test sets

```
x = df_lencoder.drop(columns=['Price (in rupees)']) y = df_lencoder['Price (in
rupees)'] x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=1) print('jumlah data', len(x)) print('')
print('jumlah data latih', len(x_train)) print('') print('jumlah data tes',
len(x_test))
```

STEP 18: Trains three different regression models to predict house prices

```
model_1 = linear_model.Lars(n_nonzero_coefs=1).fit(x_train, y_train)

model_2 = LinearRegression().fit(x_train, y_train)

model_3 = GradientBoostingRegressor(random_state=184) model_3.fit(x_train,
y_train)
```

STEP 19: Evaluates the performance of the LARS regression model (model_1) and presents the metrics in a DataFrame

```
predict_model1 = model_1.predict(x_test) mae_model1 =
mean_absolute_error(y_test, predict_model1) mse_model1 =
mean_squared_error(y_test, predict_model1) r2_model1 = r2_score(y_test,
predict_model1)

data = { 'MAE' : [mae_model1], 'MSE' : [mse_model1], 'R2' : [r2_model1] }

df_results = pd.DataFrame(data, index=['model1_LAR']) df_results
```

STEP 20: Evaluates the Linear Regression model (model_2) and adds its performance metrics to the results DataFrame

```
predict_model2 = model_2.predict(x_test) mae_model2 =
mean_absolute_error(y_test, predict_model2) mse_model2 =
mean_squared_error(y_test, predict_model2) r2_model2 = r2_score(y_test,
predict_model2)

df_results.loc['model2_LR'] = [mae_model2, mse_model2, r2_model2] df_results
```

STEP 21: Evaluates the Gradient Boosting Regressor (model_3) and adds its performance metrics to the comparison DataFrame

```
predict_model3 = model_3.predict(x_test) mae_model3 =
mean_absolute_error(y_test, predict_model3) mse_model3 =
mean_squared_error(y_test, predict_model3) r2_model3 = r2_score(y_test,
predict_model3)

df_results.loc['model3_GBR'] = [mae_model3, mse_model3, r2_model3] df_results
```