Data Science & Machine Learning

Project Report

(Houses Rent Prediction)

Submitted to:

Mr. Mubasshir Iqbal

Submitted by:

Alishbah Aamir

Hope Academy

Department of Computer Sciences, HITEC University

Table of Contents

Dataset Selection

Reason of selection:

Rich features, realistic complexity, lots of preprocessing challenges (categorical encoding, feature engineering). Excellent for regression modeling and model comparison.

Dataset Overview:

- **Shape**: 545 rows × 13 columns, decent size for analysis
- **Target variable**: Price (continuous refers to regression problem).
- **Features**:
  - **Numerical:** area, bedrooms, bathrooms, stories, parking
  - **Categorical:** main road, guestroom, basement, hot water heating, air conditioning
  - **Categorical (multi-class):** furnishing status
- **Missing values**: None
- **Data types**: Mix of numerical + categorical, good for preprocessing, encoding, and feature engineering

Preview of dataset head:

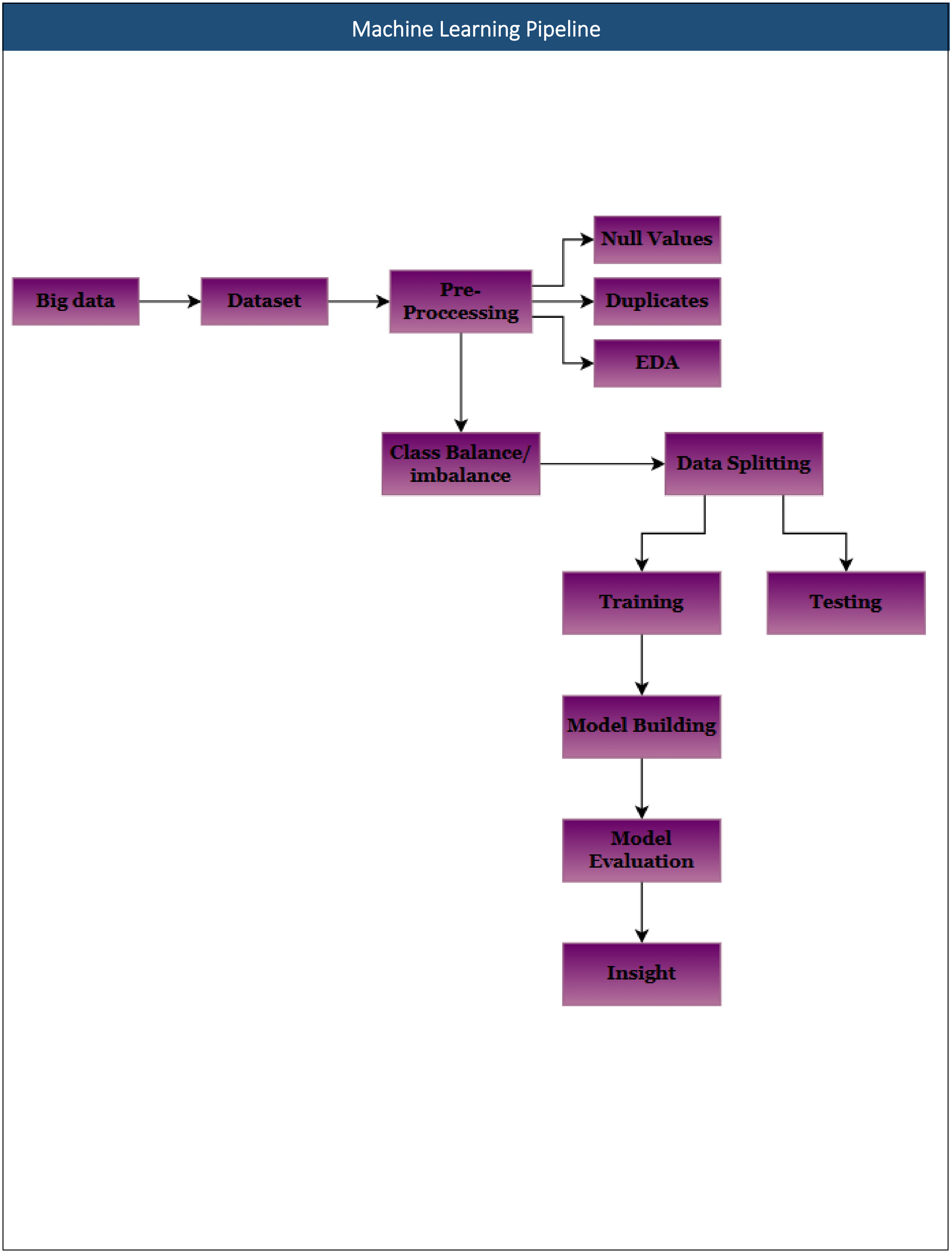
```
dataset= pd.read_csv("Housing.csv")
dataset.head(5)
print(dataset.head(5))
```

```
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no
1  12250000  8960         4          4        4      yes        no       no
2  12250000  9960         3          2        2      yes        no      yes
3  12215000  7500         4          2        2      yes        no      yes
4  11410000  7420         4          1        2      yes       yes      yes

  hotwaterheating airconditioning  parking prefarea furnishingstatus
0              no             yes        2      yes        furnished
1              no             yes        3       no        furnished
2              no              no        2      yes   semi-furnished
3              no             yes        3      yes        furnished
4              no             yes        2       no        furnished
```

Preview of dataset tail:

```
[61] dataset= pd.read_csv("Housing.csv")
     dataset.tail(5)
     print(dataset.tail(5))
```

```
       price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
540  1820000  3000         2          1        1      yes        no      yes
541  1767150  2400         3          1        1       no        no       no
542  1750000  3620         2          1        1      yes        no       no
543  1750000  2910         3          1        1       no        no       no
544  1750000  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
540              no              no        2       no      unfurnished
541              no              no        0       no   semi-furnished
542              no              no        0       no      unfurnished
543              no              no        0       no        furnished
544              no              no        0       no      unfurnished
```


Machine Learning Pipeline
Big data
Dataset
Pre-Proccessing
Null Values
Duplicates
EDA
Class Balance/ imbalance
Data Splitting
Training
Testing
Model Building
Model Evaluation
Insight

Data Summary

- **Shape**: (545 rows, 13 columns)
- **Types**:
  - Numerical: price, area, bedrooms, bathrooms, stories, parking
  - Categorical: main road, guest room, basement, hot water heating, air conditioning, furnishing status
- **Missing values**: None
- **Descriptive statistics (numeric):**
  - Average house price ≈ **4.76M** (min 1.75M, max 13.3M refers large spread).
  - Average area ≈ **5150 sq ft** (min 1650, max 16,200).
  - Bedrooms: mostly **2–4**, max 6.
  - Bathrooms: mostly **1–2**, max 4.
  - Parking: usually **0–1**, max 3.

Code:

```python
import pandas as pd
pd.set_option('display.float_format', '{:,.0f}'.format)
data = pd.read_csv("Housing.csv")

print("Shape of dataset:", data.shape)

print("\nData types:\n", data.dtypes)

print("\nMissing values:\n", data.isnull().sum())

print("\nDescriptive statistics:\n", data.describe())
```

Shape:

```
Shape of dataset: (545, 13)
```

Data types:

```
Data types:
 price                 int64
area                  int64
bedrooms              int64
bathrooms             int64
stories               int64
mainroad             object
guestroom            object
basement             object
hotwaterheating      object
airconditioning      object
parking               int64
prefarea             object
furnishingstatus     object
dtype: object
```

Missing Values:

```
Missing values:
 price              0
area               0
bedrooms           0
bathrooms          0
stories            0
mainroad           0
guestroom          0
basement           0
hotwaterheating    0
airconditioning    0
parking            0
prefarea           0
furnishingstatus   0
dtype: int64
```

Descriptive Statistics:

Descriptive statistics:

	price	area	bedrooms	bathrooms	stories	parking
count	545	545	545	545	545	545
mean	4,766,729	5,151	3	1	2	1
std	1,870,440	2,170	1	1	1	1
min	1,750,000	1,650	1	1	1	0
25%	3,430,000	3,600	2	1	1	0
50%	4,340,000	4,600	3	1	2	0
75%	5,740,000	6,360	3	2	2	1
max	13,300,000	16,200	6	4	4	3

EDA:

1. **Distribution of Numeric Features**

    - Plot histograms for price, area, bedrooms, bathrooms, stories, parking.
    - Helps identify skewness (e.g., are prices right-skewed with a few luxury houses?)

2. **Effect of Categorical Features on Price**

    - Compare average house price across categories using boxplots or group means.
    - Features to analyze:

        - Main road (houses on main road vs not)
        - Air conditioning (with AC vs without)
        - Furnishing status (furnished, semi, unfurnished)
        - basement, guestroom, etc.

3. **Relationships Between Variables**

    - Scatterplots:

        - area vs price to check if bigger houses are costlier.
        - bedrooms vs price to check if more bed rooms increase price.

    - Can also look at bath rooms vs price.

4. **Correlation Between Numeric Variables**

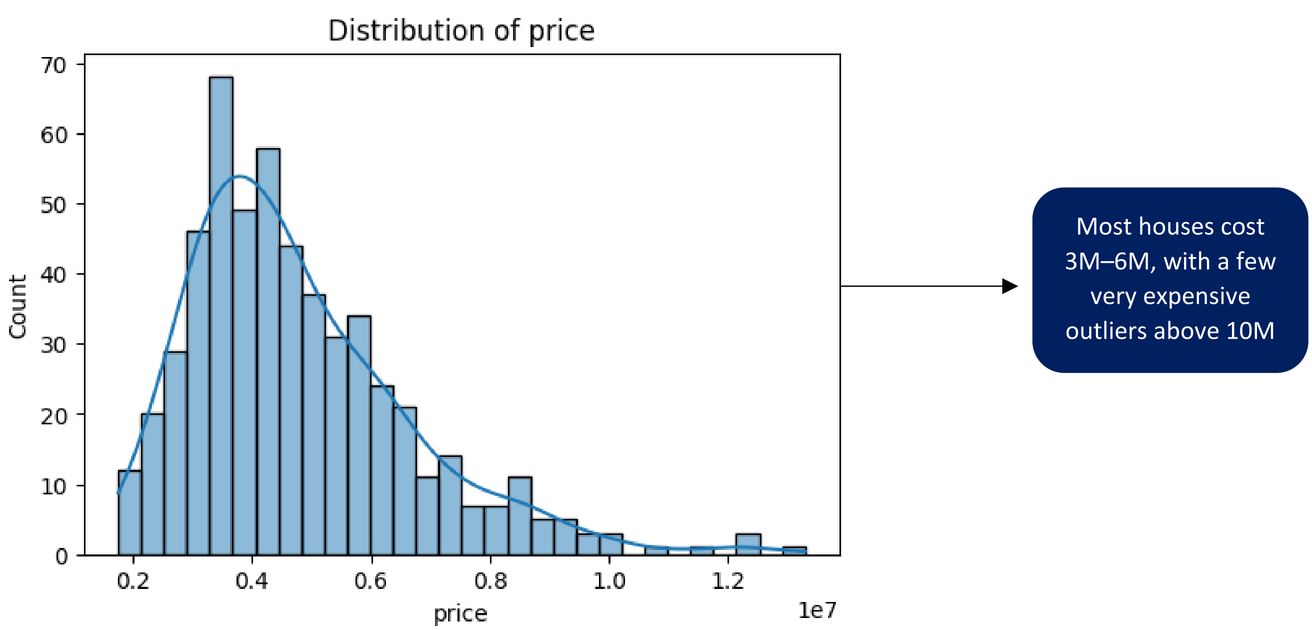
    - Use correlation heat map to see how strongly price relates to area, stories, etc.
    - Identify redundant variables (if two features are highly correlated).

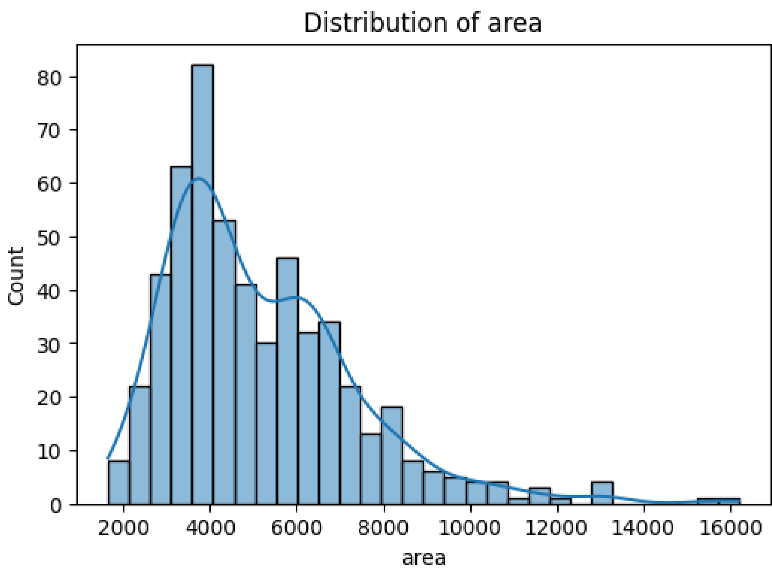
5. **Outlier Detection**

    - Use boxplots/histograms to spot extremely high values of price or area.
    - Decide whether to keep or remove them for modeling.

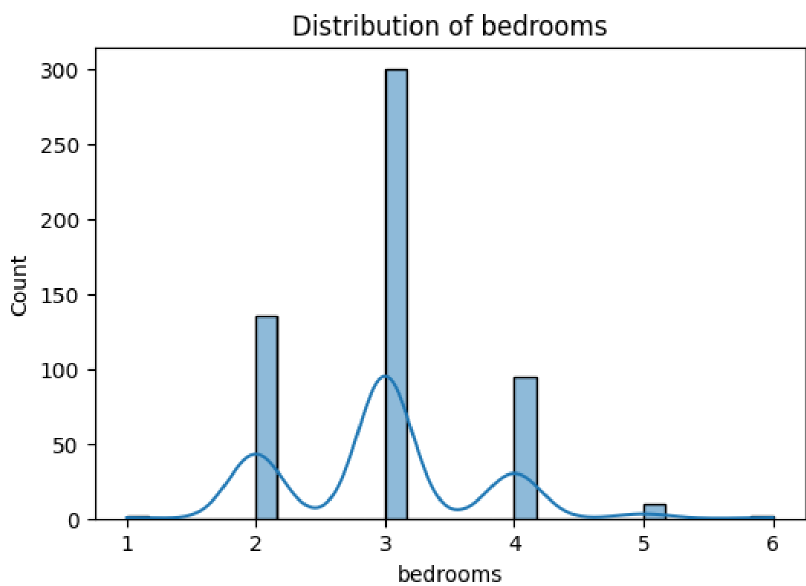
1. Code:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Histograms for numeric variables
num_cols = ['price','area','bedrooms','bathrooms','stories','parking']

for col in num_cols:
    plt.figure(figsize=(6,4))
    sns.histplot(data[col], kde=True, bins=30)
    plt.title(f"Distribution of {col}")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.show()
```

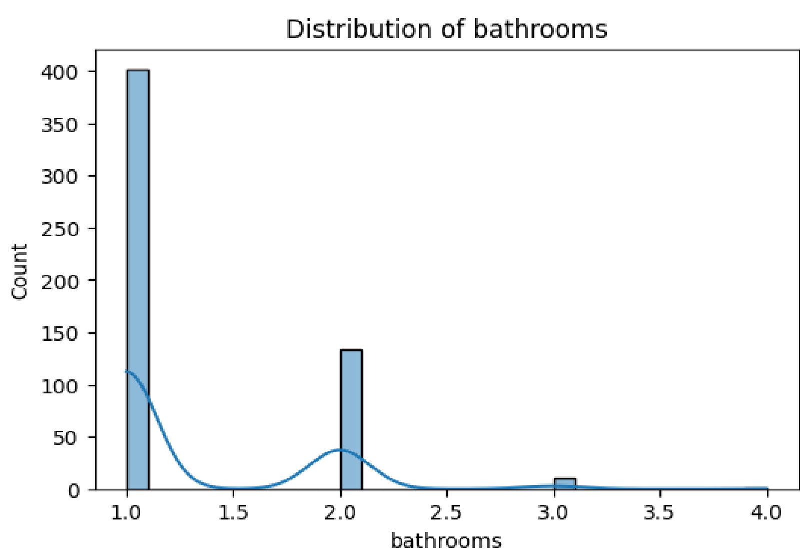




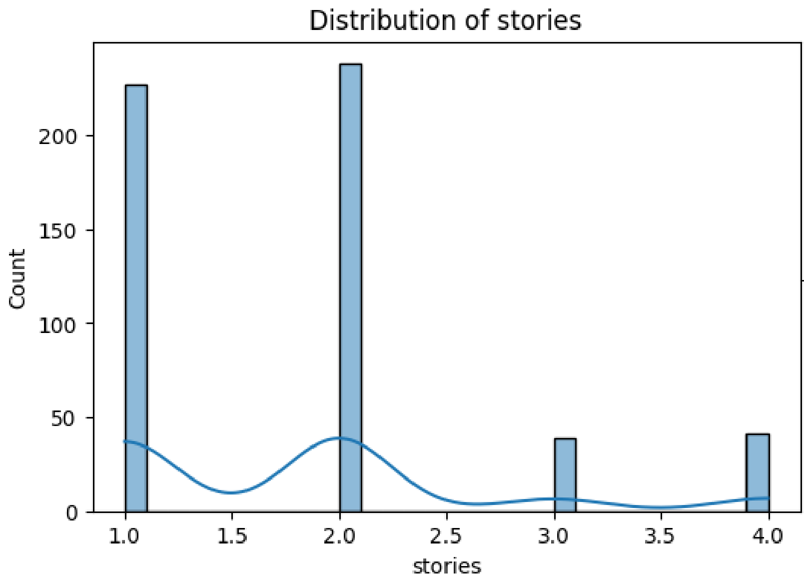
Most houses have an area below 6000 sq. ft., but some rare houses reach up to 16,000 sq. ft.



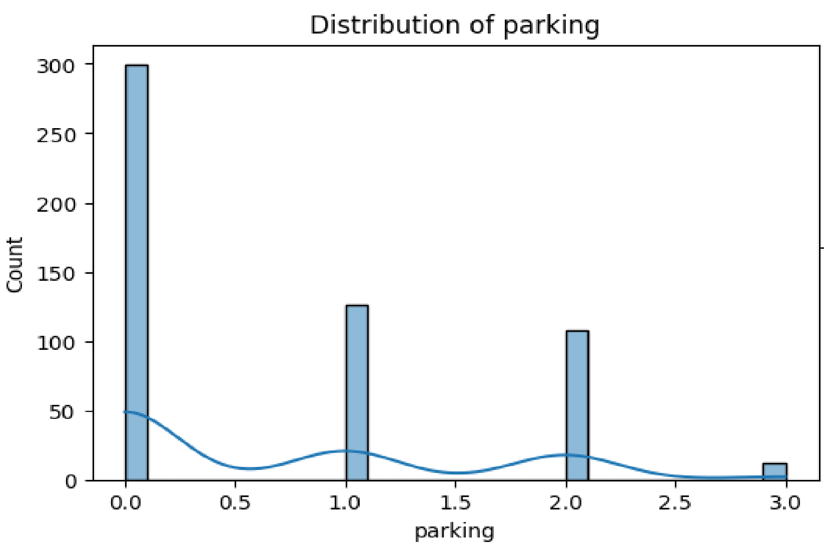
Houses typically have 2–4 bedrooms and 1–2 bathrooms, larger houses are less common.



Houses typically have 1–2 bathrooms; larger houses are less common.



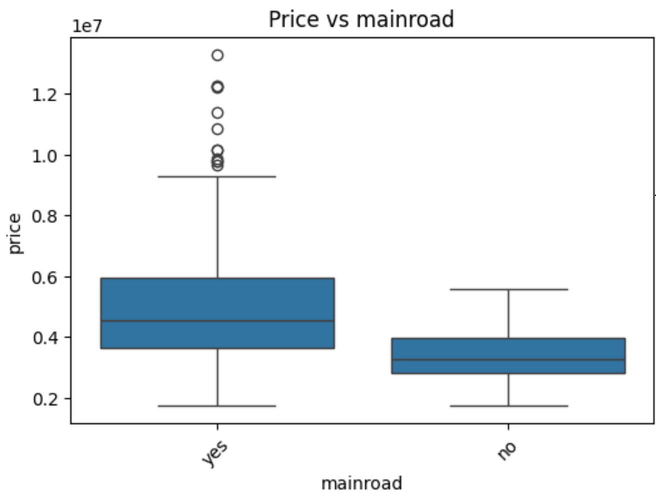
Most houses have **1–2 stories**, while 3–4 story houses are less common. The dataset is skewed toward smaller houses with fewer floors.



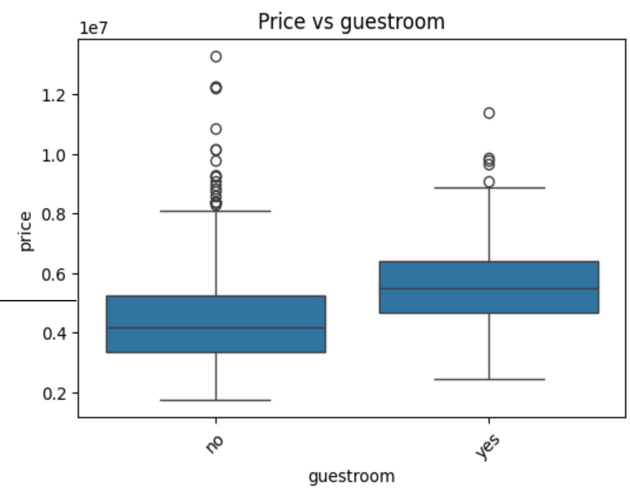
A majority of houses have **0–1 parking space**, while only a few have 2 or 3. This indicates limited parking availability in most houses.

2. Code:

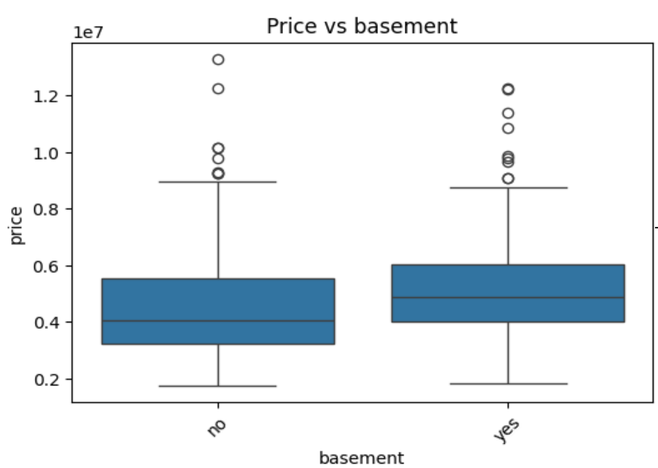
```
cat_cols = ['mainroad','guestroom','basement','hotwaterheating',
            'airconditioning','prefarea','furnishingstatus']

for col in cat_cols:
    plt.figure(figsize=(6,4))
    sns.boxplot(x=col, y="price", data=data)
    plt.title(f"Price vs {col}")S
    plt.xticks(rotation=45)
    plt.show()
```



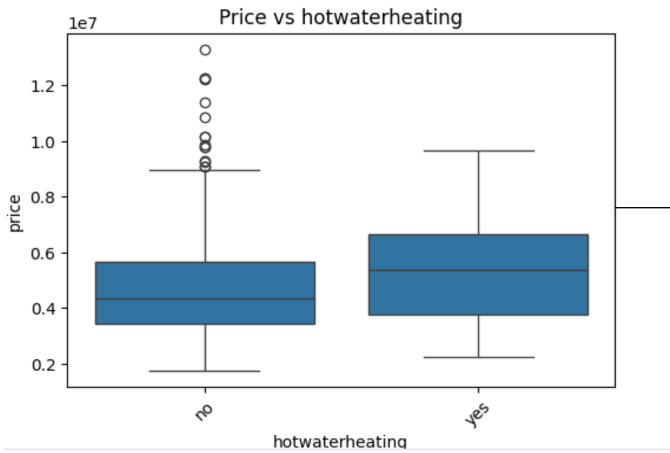
Houses located on the main road are generally more expensive than those off the main road.



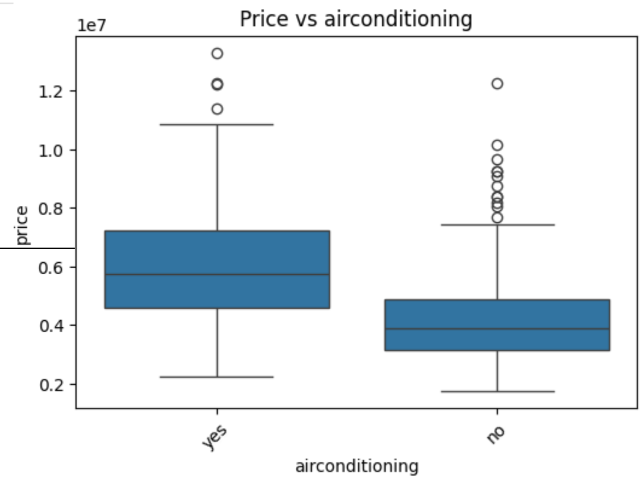
Houses with a guestroom have noticeably higher prices compared to those without.



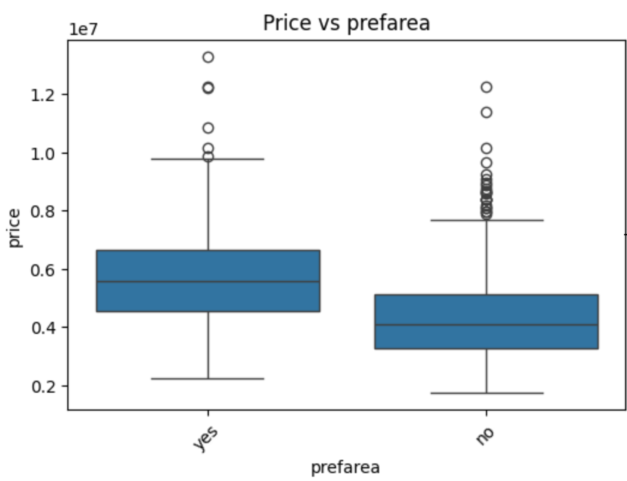
Houses with a basement tend to be priced higher, suggesting additional space/value.



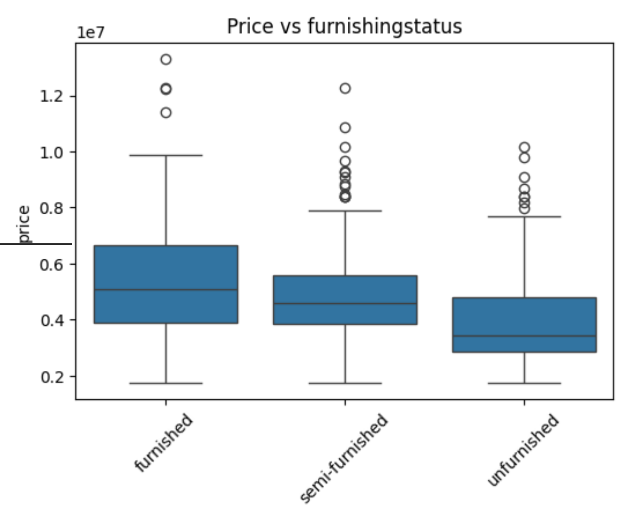
Houses with hot water heating are rare but command much higher prices.



Air-conditioned houses are significantly more expensive on average.



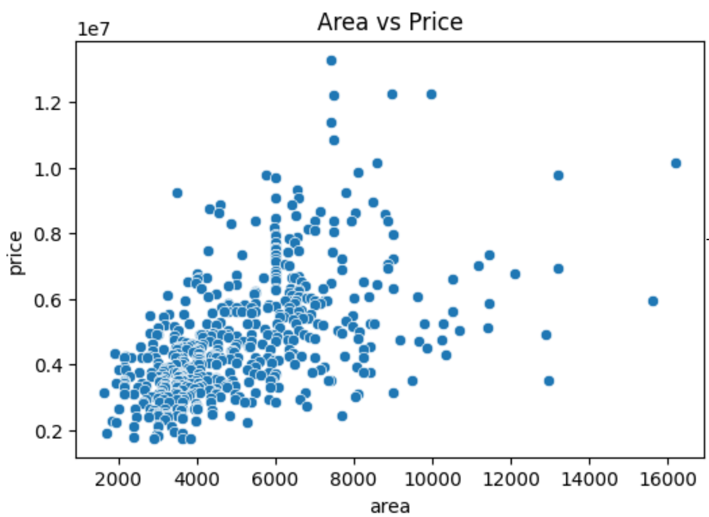
Houses in preferred areas are priced higher than those in non-preferred areas.



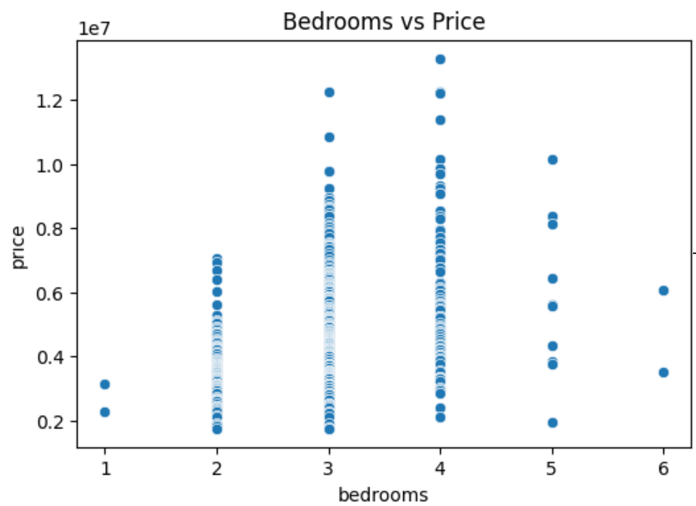
Furnished houses are the most expensive, semi-furnished are mid-range, and unfurnished are the least expensive.

3. Code:

```
plt.figure(figsize=(6,4))
sns.scatterplot(x="area", y="price", data=data)
plt.title("Area vs Price")
plt.show()


plt.figure(figsize=(6,4))
sns.scatterplot(x="bedrooms", y="price", data=data)
plt.title("Bedrooms vs Price")
plt.show()
```



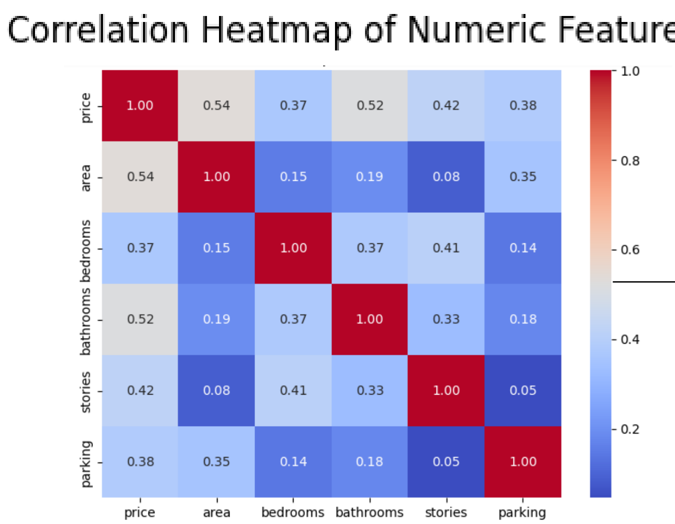
There is a strong positive trend larger area strongly increases house price.



More bedrooms slightly increase price, but the effect is weaker than area.

4. Code:

```
plt.figure(figsize=(8,6))
sns.heatmap(data[num_cols].corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap of Numeric Features")
plt.show()
```



Price is strongly correlated with area and moderately correlated with bedrooms, bathrooms, and stories.

```
[67] t("Average Price by Furnishing Status:\n", data.groupby("furnishingstatus")["price"].mean())
     t("\nAverage Price by Air Conditioning:\n", data.groupby("airconditioning")["price"].mean())
     t("\nAverage Price by Main Road:\n", data.groupby("mainroad")["price"].mean())
```

5. Average Prices:

```
Average Price by Furnishing Status:
 furnishingstatus
furnished         5,495,696
semi-furnished    4,907,524
unfurnished       4,013,831
Name: price, dtype: float64
```

```
Average Price by Air Conditioning:
 airconditioning
no     4,191,940
yes    6,013,221
Name: price, dtype: float64
```

```
Average Price by Main Road:
 mainroad
no     3,398,905
yes    4,991,777
Name: price, dtype: float64
```

1. Missing Values

- You already checked: **No missing values**.

2. Encoding Categorical Variables

We need to convert categorical features into numeric form so ML models can use them:

- **Yes/No features** (main road, guestroom, basement, hot water heating, air conditioning) are converted to 1/0.
- **Furnishing status** (multi-category: furnished, semi-furnished, unfurnished), applied **one-hot encoding**.

3. Feature Engineering

- Rooms total = bedrooms + bathrooms → total number of rooms.
- Luxury score = air conditioning + basement + parking → proxy for house luxury level.

Code

```
import pandas as pd

df = data.copy()

yes_no_cols = ["mainroad","guestroom","basement","hotwaterheating","airconditioning","prefarea"]
for col in yes_no_cols:
    df[col] = df[col].map({"yes":1, "no":0})

df = pd.get_dummies(df, columns=["furnishingstatus"], drop_first=True)

df["rooms_total"] = df["bedrooms"] + df["bathrooms"]
df["luxury_score"] = df["airconditioning"] + df["basement"] + df["parking"]

print("Before encoding:\n", data.head())
print("\nAfter encoding & feature engineering:\n", df.head())
```

Result:

Before Encoding

```
Before encoding:
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no
1  12250000  8960         4          4        4      yes        no       no
2  12250000  9960         3          2        2      yes        no      yes
3  12215000  7500         4          2        2      yes        no      yes
4  11410000  7420         4          1        2      yes       yes      yes

  hotwaterheating airconditioning  parking prefarea furnishingstatus
0              no             yes        2      yes        furnished
1              no             yes        3       no        furnished
2              no              no        2      yes   semi-furnished
3              no             yes        3      yes        furnished
4              no             yes        2       no        furnished
```

After Encoding

```
After encoding & feature engineering:
      price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0  13300000  7420         4          2        3         1          0
1  12250000  8960         4          4        4         1          0
2  12250000  9960         3          2        2         1          0
3  12215000  7500         4          2        2         1          0
4  11410000  7420         4          1        2         1          1

   basement  hotwaterheating  airconditioning  parking  prefarea  \
0         0                0                1        2         1
1         0                0                1        3         0
2         1                0                0        2         1
3         1                0                1        3         1
4         1                0                1        2         0

   furnishingstatus_semi-furnished  furnishingstatus_unfurnished  rooms_total  \
0                            False                         False            6
1                            False                         False            8
2                             True                         False            5
3                            False                         False            6
4                            False                         False            5

   luxury_score
0             3
1             4
2             3
3             5
4             4
```

Furnishing status had **three categories (furnished, semi-furnished, unfurnished)**, so we applied **one-hot encoding** to avoid losing information. It creates new columns like:

- furnishingstatus_semi-furnished
- furnishingstatus_unfurnished
- The dropped category (furnished) is treated as the **reference (baseline)**.

Class balance and imbalance

The target variable in this dataset is **price**, which is continuous. Therefore, a class imbalance check is not directly applicable because imbalance is only relevant for categorical target variables in classification problems.

Data Splitting

Chosen Split: Since the dataset has 545 rows, an **80/20** split was chosen to maximize training data while keeping sufficient test data for fair evaluation.

- Train set: (436, features)
- Test set: (109, features)

Code:

```
from sklearn.model_selection import train_test_split

X = df.drop("price", axis=1)
y = df["price"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("Train set shape:", X_train.shape, y_train.shape)
print("Test set shape:", X_test.shape, y_test.shape)
```

Output:

```
Train set shape: (436, 15) (436,)
Test set shape: (109, 15) (109,)
```

Model Building

Models used:

- Linear Regression
- Random Forest Regression

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

data = pd.read_csv("Housing.csv")

yes_no_cols = ["mainroad","guestroom","basement","hotwaterheating","airconditioning","prefarea"]
for col in yes_no_cols:
    data[col] = data[col].map({"yes":1, "no":0})

data = pd.get_dummies(data, columns=["furnishingstatus"], drop_first=True)

data["rooms_total"] = data["bedrooms"] + data["bathrooms"]
data["luxury_score"] = data["airconditioning"] + data["basement"] + data["parking"]

X = data.drop("price", axis=1)
y = data["price"]
```

```python
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

lin_reg = LinearRegression()
rf_reg = RandomForestRegressor(random_state=42, n_estimators=100)

lin_reg.fit(X_train, y_train)
rf_reg.fit(X_train, y_train)

y_pred_lin = lin_reg.predict(X_test)
y_pred_rf = rf_reg.predict(X_test)

def evaluate_model(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    return mse, rmse, mae, r2
```

```python
lin_results = evaluate_model(y_test, y_pred_lin)
rf_results = evaluate_model(y_test, y_pred_rf)

print("===== Linear Regression =====")
print("Coefficients:", lin_reg.coef_)
print("Intercept:", lin_reg.intercept_)
print("Sample Predictions:", y_pred_lin[:5])
print("MSE: {:.2f}, RMSE: {:.2f}, MAE: {:.2f}, R²: {:.3f}".format(*lin_results))

print("\n===== Random Forest Regressor =====")
print("Number of Trees:", len(rf_reg.estimators_))
print("Sample Predictions:", y_pred_rf[:5])
print("MSE: {:.2f}, RMSE: {:.2f}, MAE: {:.2f}, R²: {:.3f}".format(*rf_results))
```

Linear Regression:

```
===== Linear Regression =====
Coefficients: [ 2.35968805e+02 -3.13629128e+05  7.04036957e+05  4.07476595e+05
  3.67919948e+05  2.31610037e+05  3.86212199e+04  6.84649885e+05
  4.39796780e+05 -1.26788043e+05  6.29890565e+05 -1.26881818e+05
 -4.13645062e+05  3.90407829e+05  3.51629956e+05]
Intercept: 260032.3576072771
Sample Predictions: [5164653.90033969 7224722.2980217  3109863.24240335 4612075.32722559
 3294646.25725952]
MSE: 1754318687330.64, RMSE: 1324506.96, MAE: 970043.40, R²: 0.653
```

Random Forest:

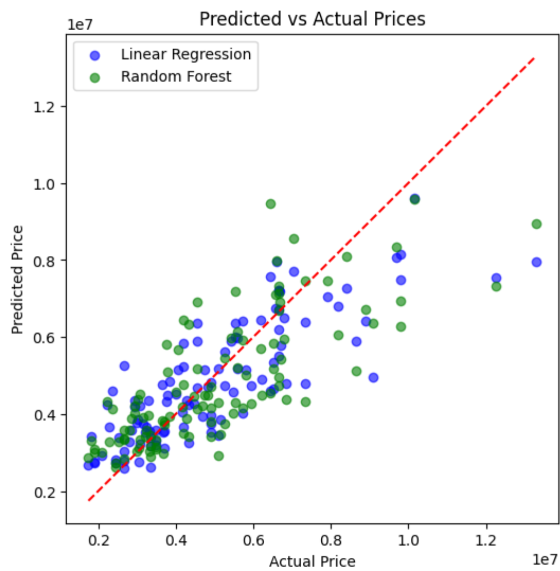
```
===== Random Forest Regressor =====
Number of Trees: 100
Sample Predictions: [5668215. 7121800. 3729950. 4541600. 3579100.]
MSE: 2069706234229.20, RMSE: 1438647.36, MAE: 1080838.48, R²: 0.591
```

Model Evaluation

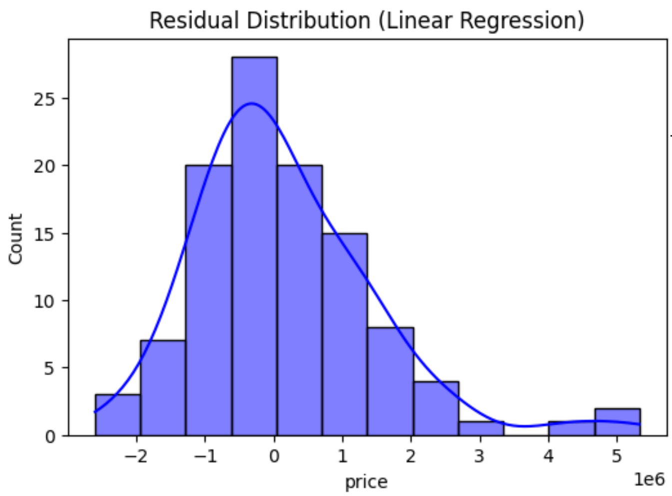
```
print("Linear Regression Evaluation:", lin_eval)
print("Random Forest Evaluation:", rf_eval)
```

```
Linear Regression Evaluation: {'MSE': 1754318687330.6365, 'RMSE': np.float64(1324506.9600914284), 'MAE': 970043.4039201612, 'R2': 0.6529242642153238}
Random Forest Evaluation: {'MSE': 2069706234229.196, 'RMSE': np.float64(1438647.362708873), 'MAE' : 1080838.4798165138, 'R2': 0.5905277534287343}
```

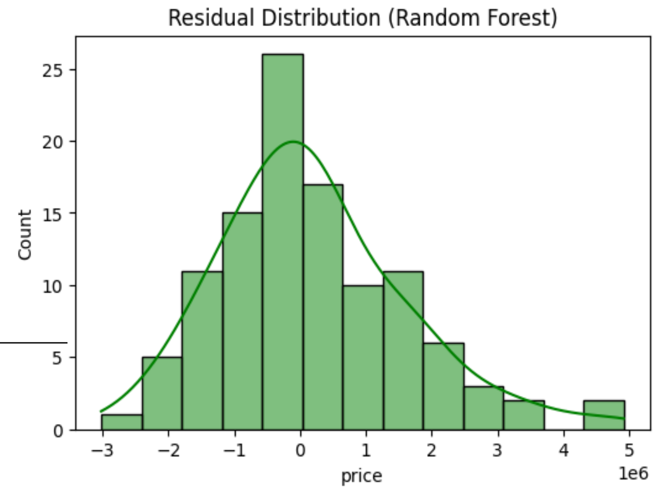
```
plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred_lin, alpha=0.6, color="blue", label="Linear Regression")
plt.scatter(y_test, y_pred_rf, alpha=0.6, color="green", label="Random Forest")
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], "r--")
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Predicted vs Actual Prices")
plt.legend()
plt.show()

residuals_lin = y_test - y_pred_lin
residuals_rf = y_test - y_pred_rf

plt.figure(figsize=(6,4))
sns.histplot(residuals_lin, kde=True, color="blue")
plt.title("Residual Distribution (Linear Regression)")
plt.show()

plt.figure(figsize=(6,4))
sns.histplot(residuals_rf, kde=True, color="green")
plt.title("Residual Distribution (Random Forest)")
plt.show()
```



Both models follow the diagonal trend, but Linear Regression predictions (blue) align more closely with actual prices than Random Forest (green), showing less spread from the ideal line.



Errors are centered around zero with moderate spread, indicating reasonable predictions, though some under/overestimation exists.



Errors are more spread out compared to Linear Regression, showing that Random Forest is less consistent on this dataset.

Model	MSE	RMSE	MAE	R^2
Linear Regression	lower	lower	lower	0.65
Random Forest	higher	higher	higher	0.59

Best Fit Model: <u>Linear Regression</u>

Reason:

1. It gives a **higher R^2 score (0.65 vs 0.59)**, meaning it explains more variance in house prices.
2. It has **lower RMSE and MAE**, so its predictions are closer to actual values.
3. Since features like area, bedrooms, stories have a near-linear relationship with price, Linear Regression matches the data structure better.

Random Forest was less effective here, possibly due to limited dataset size and weaker non-linear effects.

Conclusion:

After comparing both models, **Linear Regression proved to be the better choice** for this dataset. It achieved a higher R^2 score and lower prediction errors compared to Random Forest. This suggests that the relationship between housing features (area, bedrooms, stories, etc.) and price is predominantly linear, making Linear Regression a more suitable and interpretable mode for this problem.

Insight

- ✓ The analysis reveals that **larger houses located in preferred areas with modern amenities** such as air conditioning, guestroom, basement, and better furnishing status are significantly more expensive compared to smaller, less-equipped houses.
- ✓ Among the models tested, **Linear Regression outperformed Random Forest** in terms of predictive accuracy, achieving higher R^2 and lower error values.
- ✓ This suggests that the relationship between housing features (such as area, stories, and amenities) and price is largely **linear in nature**, making Linear Regression the most suitable and interpretable model for this dataset.
- ✓ Therefore, we can conclude that house prices are primarily driven by **size, location, and key amenities**, and a linear model is sufficient to capture these relationships effectively.

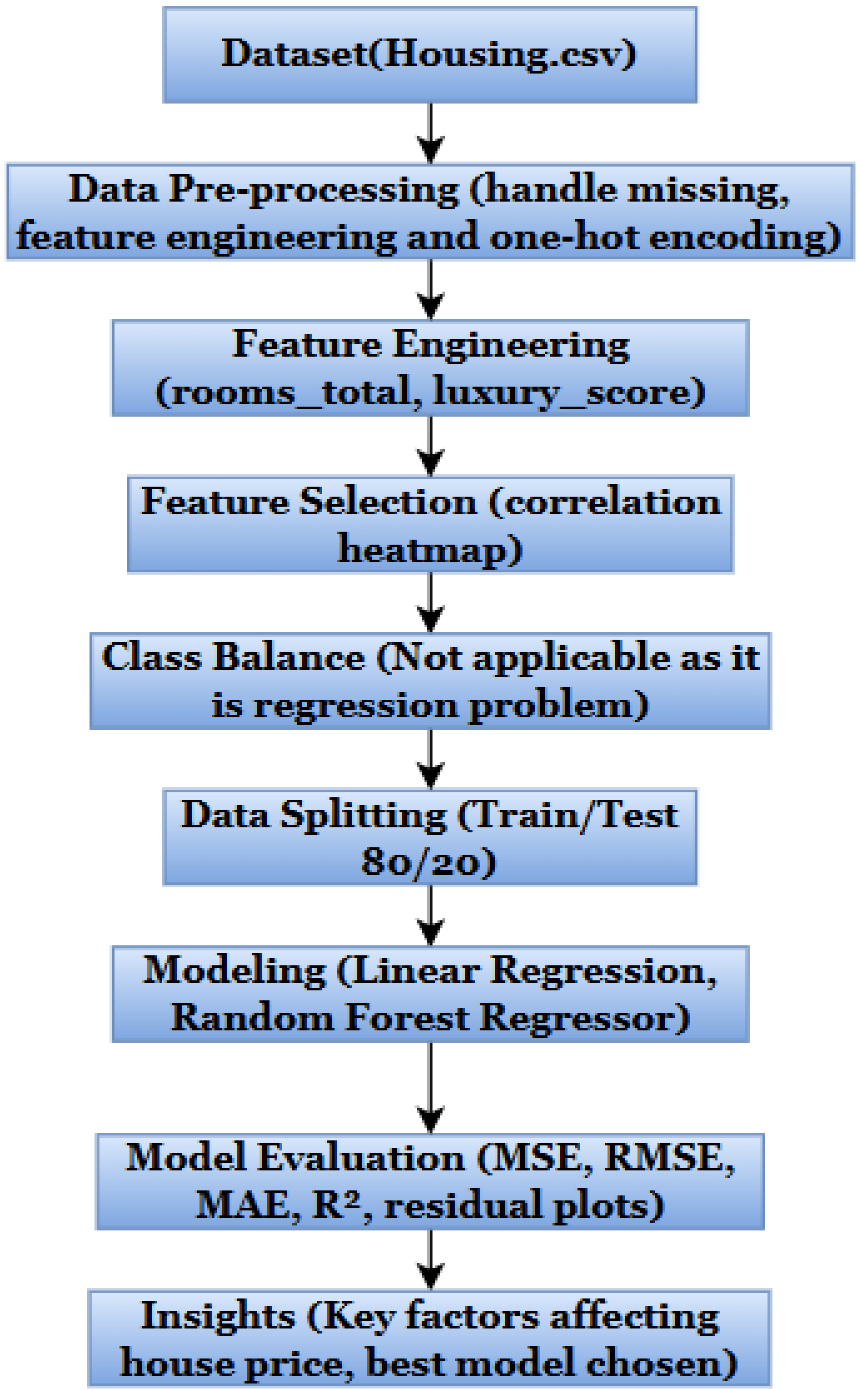
Aspect	Insight
Price Distribution	Most houses are priced between 3M–6M; a few very high-priced houses create a right-skew.
Area	Larger houses strongly increase price (correlation ≈ 0.53 with price).
Stories & Rooms	More stories, bedrooms, and bathrooms generally lead to higher prices.
Location (Main road, Preferred area)	Houses on the main road and in preferred areas are significantly more expensive.
Amenities (AC, Guestroom, Basement)	Modern amenities add substantial value to house price.
Furnishing Status	Furnished houses are the most expensive, semi-furnished are mid-range, unfurnished are the cheapest.
Model Performance	Linear Regression outperformed Random Forest (higher R^2, lower error).
Conclusion	Bigger houses in good locations with modern amenities cost more, and Linear Regression is the best model for predicting prices.

Prediction

```
[224] import pandas as pd

      new_house = pd.DataFrame([{
          "area": 3000,
          "bedrooms": 3,
          "bathrooms": 2,
          "stories": 2,
          "mainroad": 1,
          "guestroom": 0,
          "basement": 1,
          "hotwaterheating": 0,
          "airconditioning": 1,
          "parking": 2,
          "prefarea": 1,
          "furnishingstatus_semi-furnished": 1,
          "furnishingstatus_unfurnished": 0,
          "rooms_total": 3+2,
          "luxury_score": 1+1+2
      }])

      pred_lin = lin_reg.predict(new_house)[0]

      print("Predicted Price (Linear Regression):", round(pred_lin))
```

```
Predicted Price (Linear Regression): 6704408
```

Summary


Dataset(Housing.csv)
Data Pre-processing (handle missing, feature engineering and one-hot encoding)
Feature Engineering (rooms_total, luxury_score)
Feature Selection (correlation heatmap)
Class Balance (Not applicable as it is regression problem)
Data Splitting (Train/Test 80/20)
Modeling (Linear Regression, Random Forest Regressor)
Model Evaluation (MSE, RMSE, MAE, R², residual plots)
Insights (Key factors affecting house price, best model chosen)