EXPERIMENT 1

In [19]:

```
import pandas as pd

# Load the dataset using a raw string
df = pd.read_csv(r'C:\Users\hp\OneDrive\Desktop\housing.csv')
print("First 5 rows of the dataset:")
print(df.head())
```

```
First 5 rows of the dataset:
      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  
```

IMPUTATION

In [20]:

```
from sklearn.impute import SimpleImputer

# Separate numerical and categorical columns
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = df.select_dtypes(include=['object']).columns

# Impute missing values for numerical columns with mean
num_imputer = SimpleImputer(strategy='mean')
df[numerical_cols] = num_imputer.fit_transform(df[numerical_cols])

# Impute missing values for categorical columns with mode
cat_imputer = SimpleImputer(strategy='most_frequent')
df[categorical_cols] = cat_imputer.fit_transform(df[categorical_cols])

print("First 5 rows after imputation:")
print(df.head())
```

```
First 5 rows after imputation:
        price    area  bedrooms  bathrooms  stories mainroad guestroom  \
0  13300000.0  7420.0       4.0        2.0      3.0      yes        no   
1  12250000.0  8960.0       4.0        4.0      4.0      yes        no   
2  12250000.0  9960.0       3.0        2.0      2.0      yes        no   
3  12215000.0  7500.0       4.0        2.0      2.0      yes        no   
4  11410000.0  7420.0       4.0        1.0      2.0      yes       yes   

  basement hotwaterheating airconditioning  parking prefarea furnishingstatus  
0       no              no             yes      2.0      yes        furnished  
1       no              no             yes      3.0       no        furnished  
2      yes              no              no      2.0      yes   semi-furnished  
3      yes              no             yes      3.0      yes        furnished  
4      yes              no             yes      2.0       no        furnished  
```

ANOMALY DETECTION

In [25]:

```
from sklearn.ensemble import IsolationForest

# Verify the column name and use it in anomaly detection
print(df.columns)  # Print column names to identify the correct one

# Choose a numerical column for anomaly detection, e.g., 'LotArea'
# Replace 'LotArea' with the correct column name if it differs
column_to_check = 'LotArea'

# Ensure the column exists in the DataFrame
if column_to_check in df.columns:
    clf = IsolationForest(contamination=0.05)
    df['anomaly'] = clf.fit_predict(df[[column_to_check]])
    # Display rows marked as anomalies
    anomalies = df[df['anomaly'] == -1]
    print("Anomalous rows:")
    print(anomalies.head())
else:
    print(f"Column '{column_to_check}' not found in DataFrame.")
```

```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'furnishingstatus'],
      dtype='object')
Column 'LotArea' not found in DataFrame.
```

STANDARDIZATION

In [22]:

```
from sklearn.preprocessing import StandardScaler

# Standardize numerical columns
scaler = StandardScaler()
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])

print("First 5 rows after standardization:")
print(df.head())
```

```
First 5 rows after standardization:
      price      area  bedrooms  bathrooms   stories mainroad guestroom  \
0  4.566365  1.046726  1.403419   1.421812  1.378217      yes        no   
1  4.004484  1.757010  1.403419   5.405809  2.532024      yes        no   
2  4.004484  2.218232  0.047278   1.421812  0.224410      yes        no   
3  3.985755  1.083624  1.403419   1.421812  0.224410      yes        no   
4  3.554979  1.046726  1.403419  -0.570187  0.224410      yes       yes   

  basement hotwaterheating airconditioning   parking prefarea furnishingstatus
0       no              no             yes  1.517692      yes        furnished
1       no              no             yes  2.679409       no        furnished
2      yes              no              no  1.517692      yes   semi-furnished
3      yes              no             yes  2.679409      yes        furnished
4      yes              no             yes  1.517692       no        furnished
```

NORMALIZATION

In [23]:

```
from sklearn.preprocessing import MinMaxScaler

# Normalize numerical columns
normalizer = MinMaxScaler()
df[numerical_cols] = normalizer.fit_transform(df[numerical_cols])

print("First 5 rows after normalization:")
print(df.head())
```

```
First 5 rows after normalization:
      price      area  bedrooms  bathrooms   stories mainroad guestroom  \
0  1.000000  0.396564       0.6   0.333333  0.666667      yes        no   
1  0.909091  0.502405       0.6   1.000000  1.000000      yes        no   
2  0.909091  0.571134       0.4   0.333333  0.333333      yes        no   
3  0.906061  0.402062       0.6   0.333333  0.333333      yes        no   
4  0.836364  0.396564       0.6   0.000000  0.333333      yes       yes   

  basement hotwaterheating airconditioning   parking prefarea furnishingstatus
0       no              no             yes  0.666667      yes        furnished
1       no              no             yes  1.000000       no        furnished
2      yes              no              no  0.666667      yes   semi-furnished
3      yes              no             yes  1.000000      yes        furnished
4      yes              no             yes  0.666667       no        furnished
```

ENCODING

In [26]:

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Load the dataset
df = pd.read_csv(r'C:\Users\hp\OneDrive\Desktop\housing.csv')  # Update the path if necessary

# Strip leading and trailing spaces from column names, if any
df.columns = df.columns.str.strip()

# Separate categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
```

```
# Initialize the OneHotEncoder with sparse_output=False
encoder = OneHotEncoder(sparse_output=False, drop='first')  # Use sparse_output instead of sparse

# Fit and transform the categorical columns
encoded_features = encoder.fit_transform(df[categorical_cols])

# Create DataFrame for encoded features and concatenate with original DataFrame
encoded_df = pd.DataFrame(encoded_features, columns=encoder.get_feature_names_out(categorical_cols))
df_encoded = pd.concat([df.drop(categorical_cols, axis=1), encoded_df], axis=1)

print("First 5 rows after encoding:")
print(df_encoded.head())
```

```
First 5 rows after encoding:
      price  area  bedrooms  bathrooms  stories  parking  mainroad_yes  \
0  13300000  7420         4          2        3        2           1.0   
1  12250000  8960         4          4        4        3           1.0   
2  12250000  9960         3          2        2        2           1.0   
3  12215000  7500         4          2        2        3           1.0   
4  11410000  7420         4          1        2        2           1.0   

   guestroom_yes  basement_yes  hotwaterheating_yes  airconditioning_yes  \
0            0.0           0.0                  0.0                  1.0   
1            0.0           0.0                  0.0                  1.0   
2            0.0           1.0                  0.0                  0.0   
3            0.0           1.0                  0.0                  1.0   
4            1.0           1.0                  0.0                  1.0   

   prefarea_yes  furnishingstatus_semi-furnished  furnishingstatus_unfurnished
0           1.0                              0.0                           0.0
1           0.0                              0.0                           0.0
2           1.0                              1.0                           0.0
3           1.0                              0.0                           0.0
4           0.0                              0.0                           0.0
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js