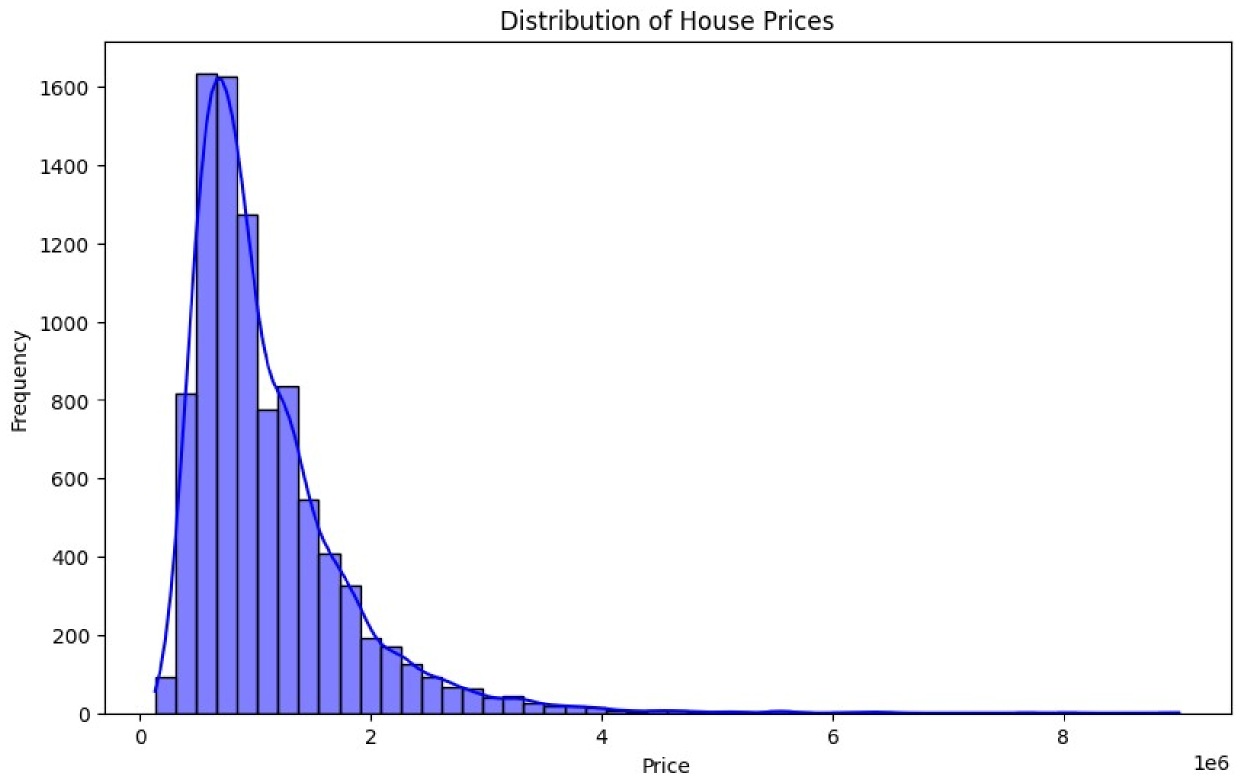
```python
# Import necessary libraries
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error

# Suppress warnings for cleaner output
warnings.filterwarnings('ignore')

# Load the dataset
print("Loading dataset...")
try:
    df =
pd.read_csv("/kaggle/input/melbourne-house-pricing/Melbourne_housing_F
ULL.csv")  # Replace with your dataset path
    print(f"Dataset loaded with {df.shape[0]} rows and {df.shape[1]}
columns.\n")
except FileNotFoundError:
    print("Error: The file 'Melbourne_housing_FULL.csv' was not found.
Please check the file path.")
    exit()

# Step 1: Data Cleaning
print("Cleaning the data...")
# Drop columns that are not relevant for predictions
columns_to_drop = ['Address', 'Date', 'Postcode', 'YearBuilt',
'Lattitude', 'Longtitude']
df_cleaned = df.drop(columns=columns_to_drop, errors='ignore')

# Drop rows with missing values to handle NaNs
df_cleaned = df_cleaned.dropna()
print(f"After cleaning, dataset has {df_cleaned.shape[0]} rows and
{df_cleaned.shape[1]} columns.\n")

# Step 2: Exploratory Data Analysis (EDA)
print("Performing exploratory data analysis...")

# Distribution of 'Price'
plt.figure(figsize=(10, 6))
sns.histplot(df_cleaned['Price'], bins=50, kde=True, color='blue')
plt.title('Distribution of House Prices')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```

```
# Correlation Heatmap
print("Generating correlation heatmap...")
plt.figure(figsize=(12, 8))

# Select only numeric columns for correlation
numeric_cols = df_cleaned.select_dtypes(include=[np.number])
corr_matrix = numeric_cols.corr()

sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm",
cbar=True)
plt.title('Correlation Heatmap')
plt.show()

# Boxplot of Prices by Regionname
if 'Regionname' in df_cleaned.columns:
    plt.figure(figsize=(12, 6))
    sns.boxplot(x='Regionname', y='Price', data=df_cleaned,
palette='Set3')
    plt.title('Boxplot of Prices by Region')
    plt.xlabel('Region')
    plt.ylabel('Price')
    plt.xticks(rotation=45)
    plt.show()

# Pairplot of key numeric features
numeric_features = ['Price', 'Rooms', 'Distance', 'Landsize',
'BuildingArea']
available_features = [col for col in numeric_features if col in
df_cleaned.columns]
sns.pairplot(df_cleaned[available_features], diag_kind='kde',
markers='o')
plt.suptitle('Pairplot of Key Numeric Features', y=1.02)
plt.show()

# Step 3: Feature Engineering
print("\nEngineering features...")
df_encoded = pd.get_dummies(df_cleaned, drop_first=True, dtype=int)
print(f"Dataset after encoding has {df_encoded.shape[1]} columns.\n")

# Step 4: Define Features and Target
if 'Price' not in df_encoded.columns:
    print("Error: 'Price' column is missing in the dataset.")
    exit()
X = df_encoded.drop(columns=['Price'])
Y = df_encoded['Price']

# Step 5: Train-Test Split
print("Splitting dataset into training and testing sets...")
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.3, random_state=42)
```

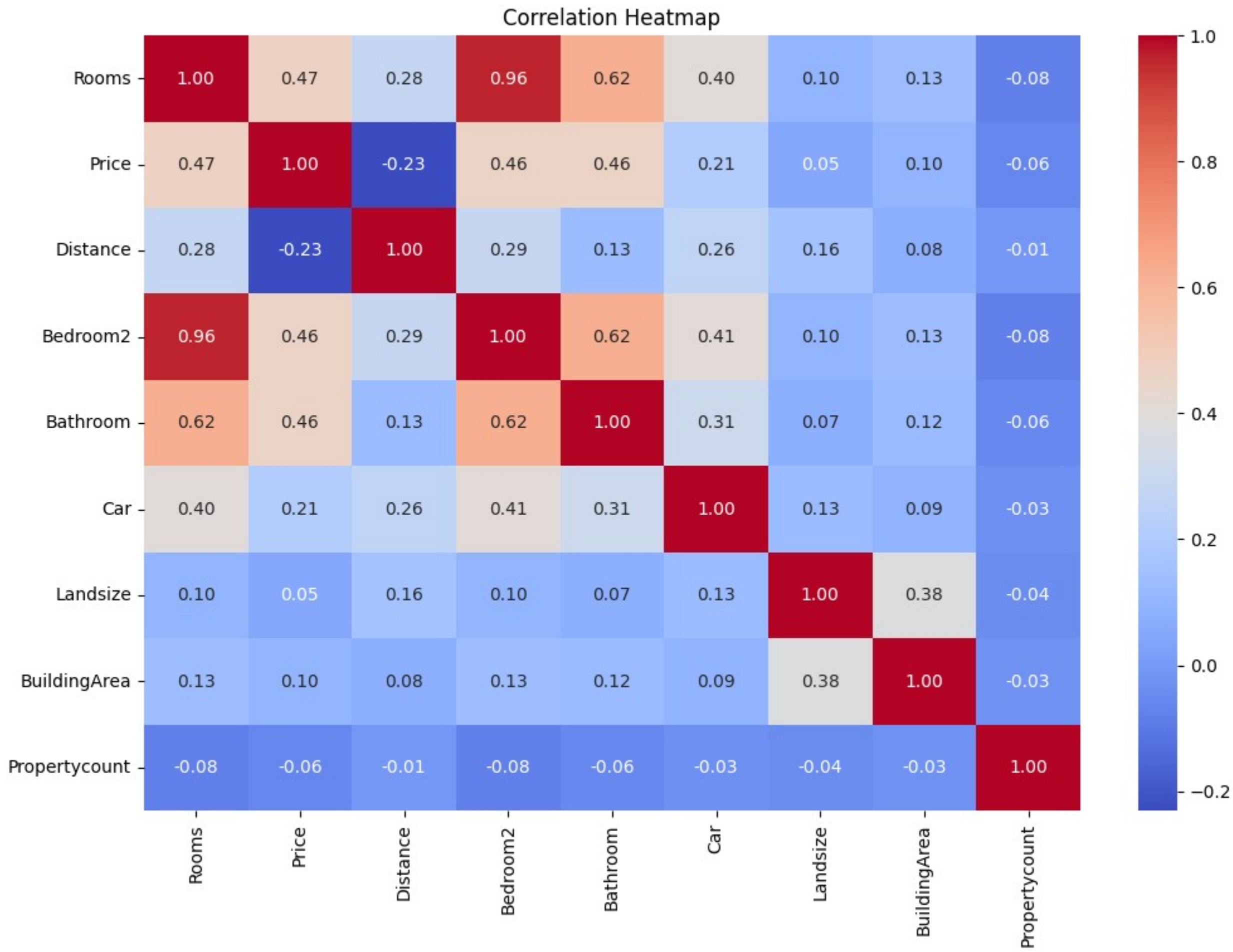
```python
print(f"Training set: {X_train.shape[0]} samples, Testing set:
{X_test.shape[0]} samples.\n")

# Step 6: Model Training and Evaluation - Linear Regression
print("Training Linear Regression model...")
linear_model = LinearRegression()
linear_model.fit(X_train, Y_train)

# Predictions and evaluation
linear_train_preds = linear_model.predict(X_train)
linear_test_preds = linear_model.predict(X_test)
linear_train_r2 = r2_score(Y_train, linear_train_preds)
linear_test_r2 = r2_score(Y_test, linear_test_preds)

print(f"Linear Regression - Training R^2: {linear_train_r2:.4f}")
print(f"Linear Regression - Testing R^2: {linear_test_r2:.4f}\n")

# Scatter plot: Actual vs Predicted for Linear Regression
plt.figure(figsize=(10, 6))
plt.scatter(Y_test, linear_test_preds, alpha=0.5, color='red')
plt.plot([min(Y_test), max(Y_test)], [min(Y_test), max(Y_test)],
color='blue', lw=2)
plt.title('Linear Regression: Actual vs Predicted Prices')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.show()

# Step 7: Model Training and Evaluation - Lasso Regression
print("Training Lasso Regression model...")
lasso_model = Lasso(alpha=25, max_iter=100, tol=0.1)
lasso_model.fit(X_train, Y_train)

# Predictions and evaluation
lasso_train_preds = lasso_model.predict(X_train)
lasso_test_preds = lasso_model.predict(X_test)
lasso_train_r2 = r2_score(Y_train, lasso_train_preds)
lasso_test_r2 = r2_score(Y_test, lasso_test_preds)

print(f"Lasso Regression - Training R^2: {lasso_train_r2:.4f}")
print(f"Lasso Regression - Testing R^2: {lasso_test_r2:.4f}\n")

# Scatter plot: Actual vs Predicted for Lasso Regression
plt.figure(figsize=(10, 6))
plt.scatter(Y_test, lasso_test_preds, alpha=0.5, color='green')
plt.plot([min(Y_test), max(Y_test)], [min(Y_test), max(Y_test)],
color='blue', lw=2)
plt.title('Lasso Regression: Actual vs Predicted Prices')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.show()
```

```
# Step 8: Display Additional Metrics
print("\nEvaluating models with additional metrics...")
print("\nLinear Regression:")
print(f"Mean Absolute Error: {mean_absolute_error(Y_test,
linear_test_preds):.2f}")
print(f"Mean Squared Error: {mean_squared_error(Y_test,
linear_test_preds):.2f}")
print(f"Root Mean Squared Error: {np.sqrt(mean_squared_error(Y_test,
linear_test_preds)):.2f}")

print("\nLasso Regression:")
print(f"Mean Absolute Error: {mean_absolute_error(Y_test,
lasso_test_preds):.2f}")
print(f"Mean Squared Error: {mean_squared_error(Y_test,
lasso_test_preds):.2f}")
print(f"Root Mean Squared Error: {np.sqrt(mean_squared_error(Y_test,
lasso_test_preds)):.2f}")

# Save the cleaned dataset
df_encoded.to_csv("Cleaned_Melbourne_Housing_Data.csv", index=False)
print("\nCleaned and encoded dataset saved as
'Cleaned_Melbourne_Housing_Data.csv'.")
```

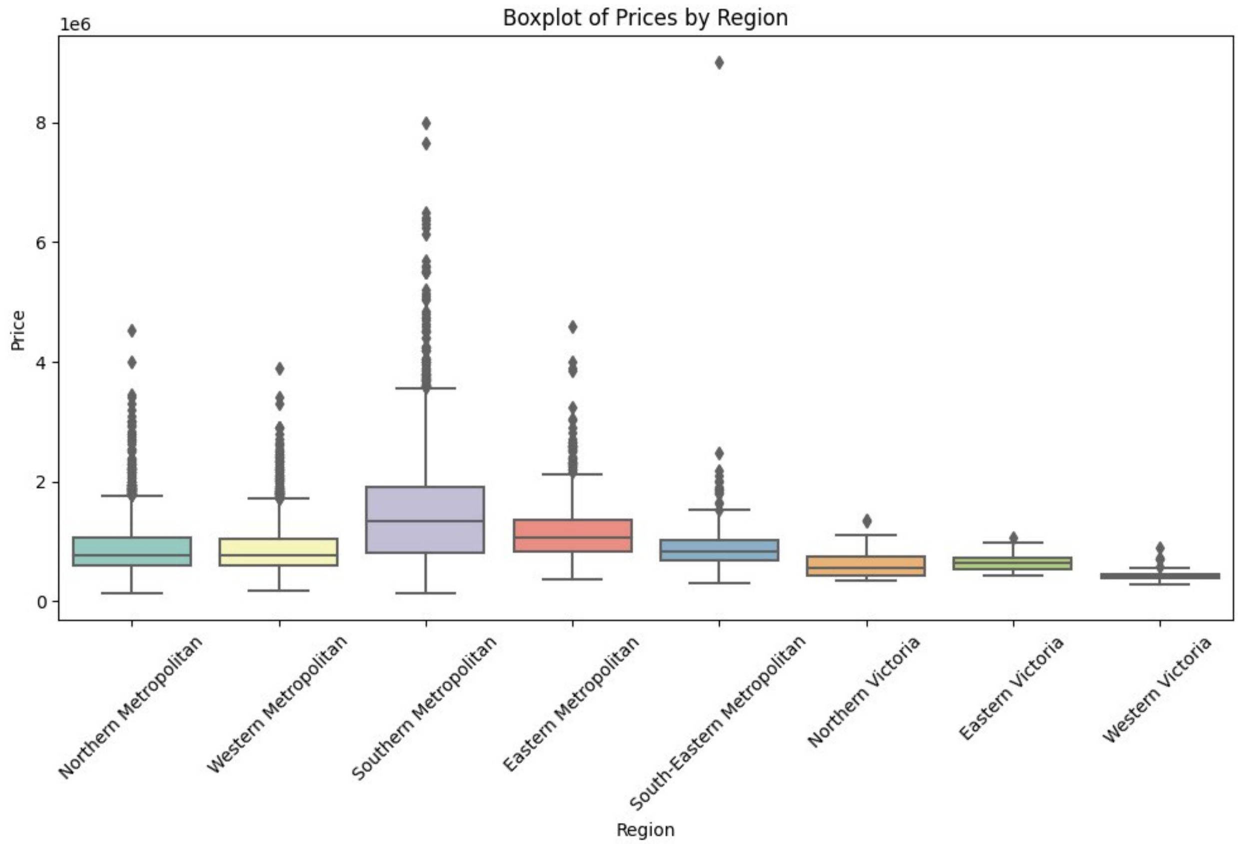
```
Loading dataset...
Dataset loaded with 34857 rows and 21 columns.

Cleaning the data...
After cleaning, dataset has 9244 rows and 15 columns.

Performing exploratory data analysis...
```



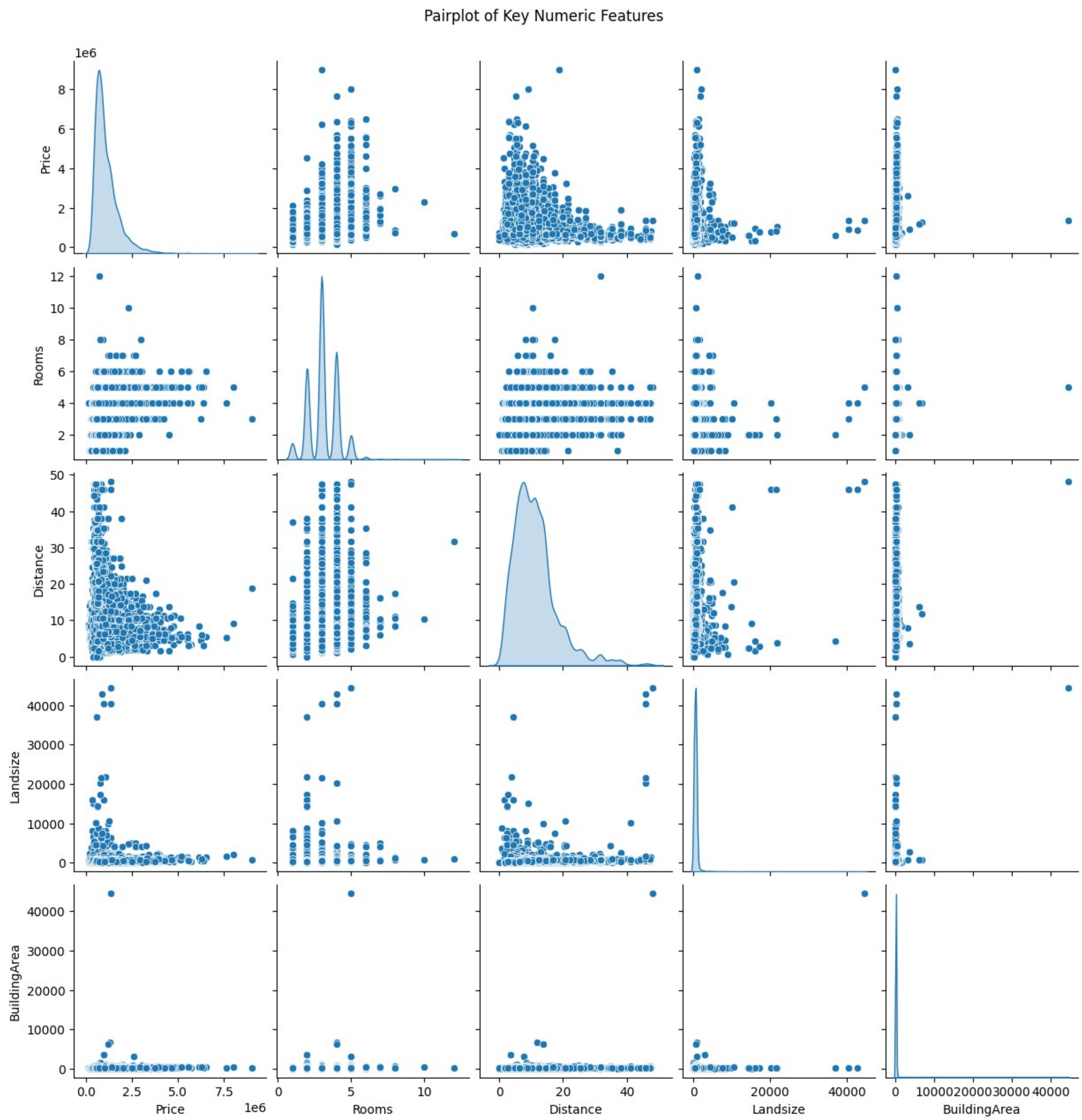

```
Generating correlation heatmap...
```



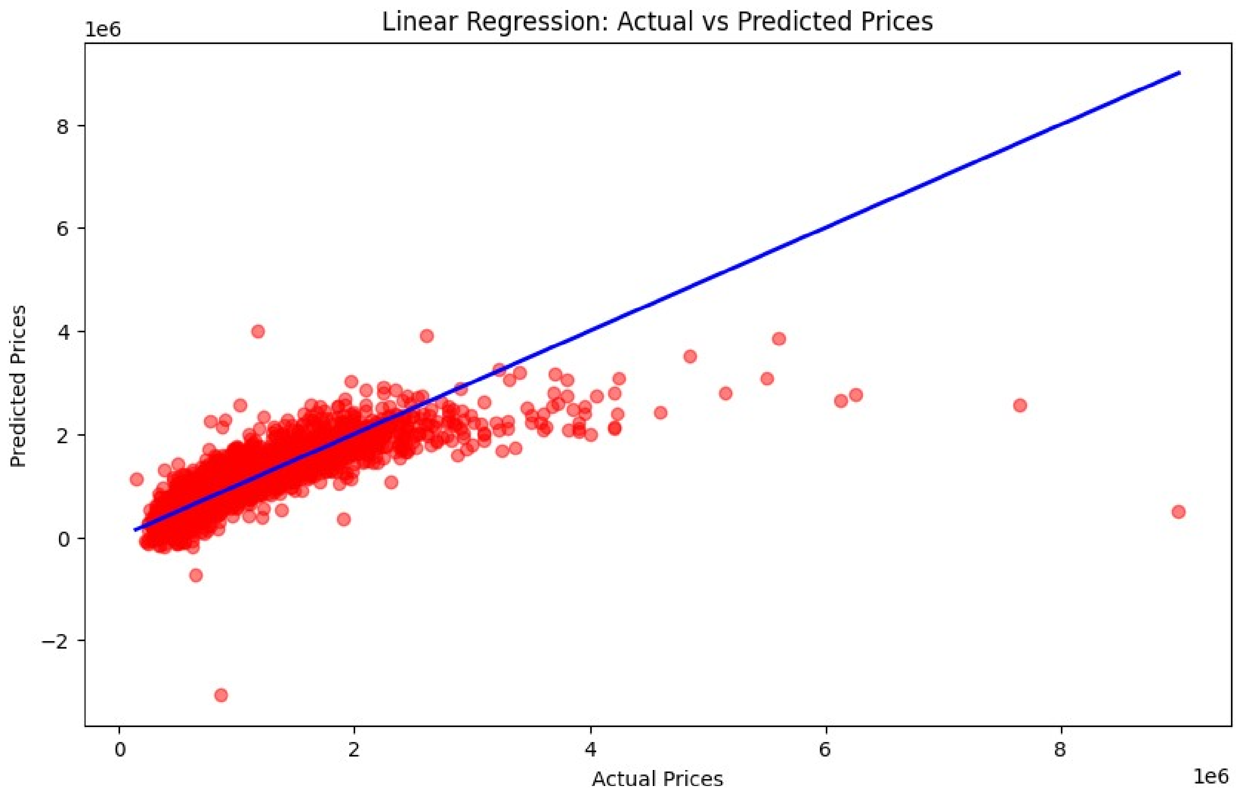
Correlation Heatmap
Rooms
Price
Distance
Bedroom2
Bathroom
Car
Landsize
BuildingArea
Propertycount
1.00 0.47 0.28 0.96 0.62 0.40 0.10 0.13 -0.08
0.47 1.00 -0.23 0.46 0.46 0.21 0.05 0.10 -0.06
0.28 -0.23 1.00 0.29 0.13 0.26 0.16 0.08 -0.01
0.96 0.46 0.29 1.00 0.62 0.41 0.10 0.13 -0.08
0.62 0.46 0.13 0.62 1.00 0.31 0.07 0.12 -0.06
0.40 0.21 0.26 0.41 0.31 1.00 0.13 0.09 -0.03
0.10 0.05 0.16 0.10 0.07 0.13 1.00 0.38 -0.04
0.13 0.10 0.08 0.13 0.12 0.09 0.38 1.00 -0.03
-0.08 -0.06 -0.01 -0.08 -0.06 -0.03 -0.04 -0.03 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



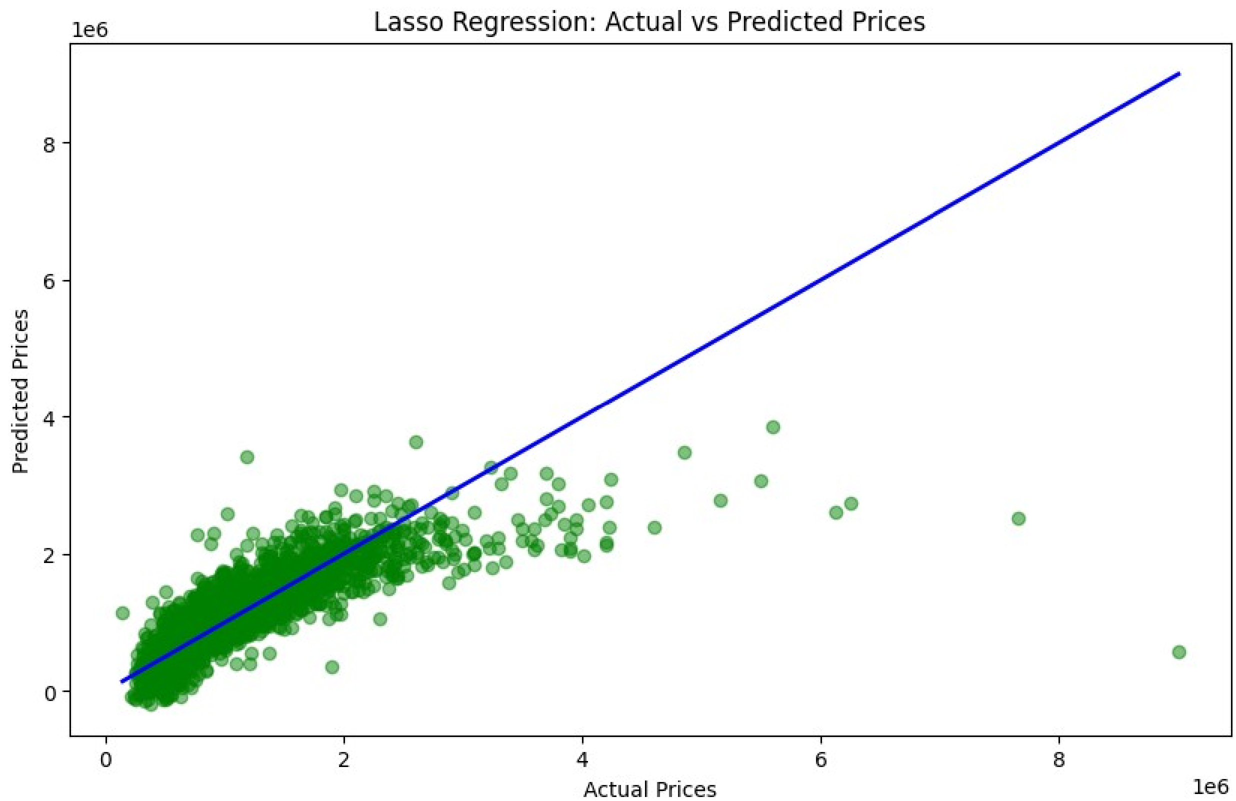
Boxplot of Prices by Region
1e6
8
6
4
2
0
Price
Northern Metropolitan
Western Metropolitan
Southern Metropolitan
Eastern Metropolitan
South-Eastern Metropolitan
Northern Victoria
Eastern Victoria
Western Victoria
Region




```
Engineering features...
Dataset after encoding has 624 columns.

Splitting dataset into training and testing sets...
Training set: 6470 samples, Testing set: 2774 samples.

Training Linear Regression model...
Linear Regression - Training R^2: 0.7491
Linear Regression - Testing R^2: 0.6415
```




```
Training Lasso Regression model...
Lasso Regression - Training R^2: 0.7475
Lasso Regression - Testing R^2: 0.6620
```




```
Evaluating models with additional metrics...

Linear Regression:
Mean Absolute Error: 241722.80
Mean Squared Error: 168143810173.95
Root Mean Squared Error: 410053.42

Lasso Regression:
Mean Absolute Error: 235977.93
Mean Squared Error: 158527138513.74
Root Mean Squared Error: 398154.67

Cleaned and encoded dataset saved as
'Cleaned_Melbourne_Housing_Data.csv'.
```