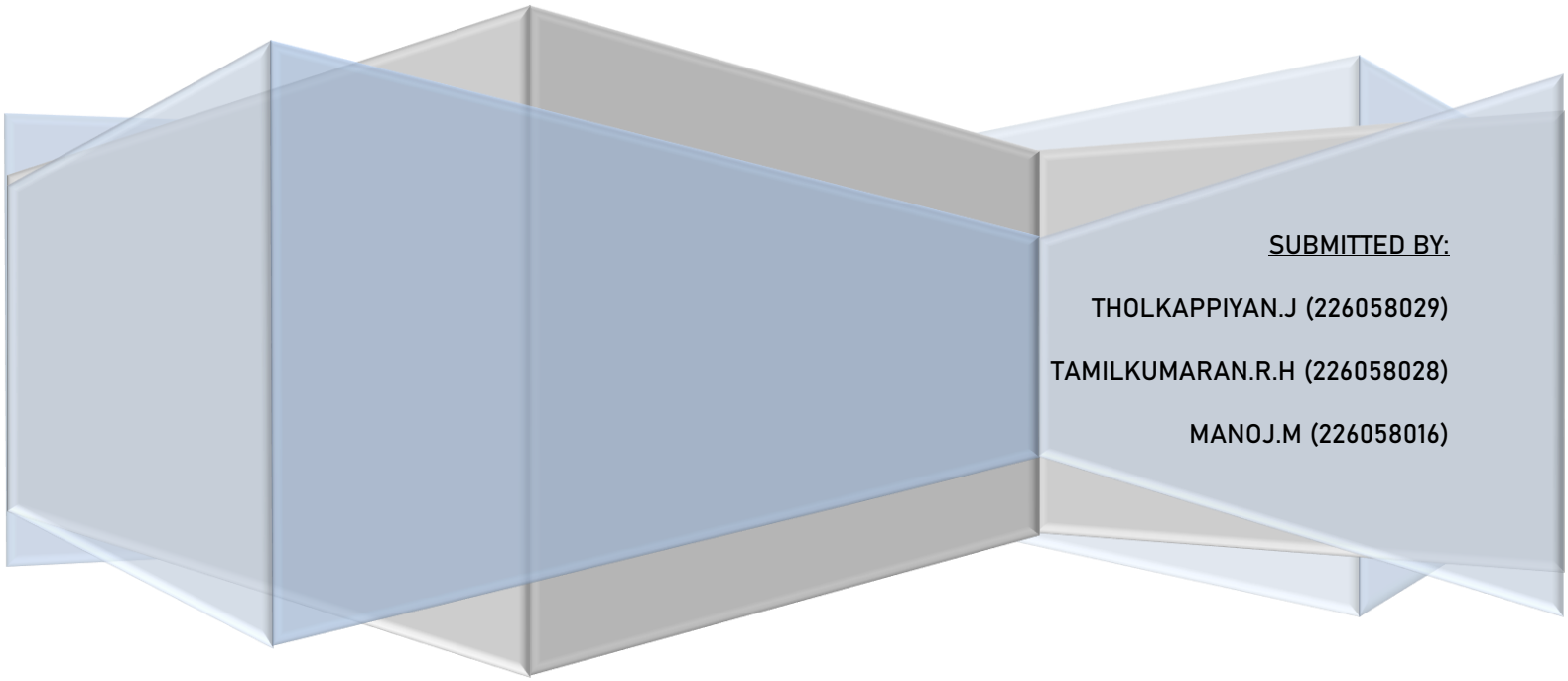
SRC CAMPUS – SASTRA DEEMED TO BE UNIVERSITY
DEPARTMENT : M.SC.COMPUTER SCIENCE(IST YEAR – A)
COURSE : PYTHON & DATA ANALYTICS
COURSE CODE: CSS423

HOUSE PRICE PREDICTION ANALYSIS

(DATA EXPLORATION AND MODEL EVALUATION)



SUBMITTED BY:

THOLKAPPIYAN.J (226058029)

TAMILKUMARAN.R.H (226058028)

MANOJ.M (226058016)

ABSTRACT:

- **This dataset consists of 2,000 rows of house-related data, capturing a variety of features that influence house prices.**

- **Key attributes include area, number of bedrooms and bathrooms, number of floors, year built, location, condition, and the presence of a garage, with the target variable being the sale price of the house.**

- **These features offer valuable insights for predicting house prices and understanding the factors that contribute to property value.**

- **The dataset covers a broad range of properties, from small apartments to luxury homes, across different locations and conditions, allowing for analysis of how these variables interact.**

- **Notably, features such as area, location, condition, and age of the house are expected to show strong correlations with price.**

- **The dataset is well-suited for various analytical applications, including house price prediction using regression techniques, feature importance analysis, clustering for market segmentation, and time-based analysis of housing trends.**

- **It provides a rich foundation for both machine learning models and exploratory data analysis aimed at understanding the dynamics of the housing market.**

INTRODUCTION:

- **Objective: Analyze house price data to uncover insights and build predictive models.**
- **The dataset contains 2000 rows of house-related data, representing various features that could influence house prices. Below, we discuss key aspects of the dataset, which include its structure, the choice of features, and potential use cases for analysis.**
- **The dataset is designed to capture essential attributes for predicting house prices, including:**
- **Area: Square footage of the house, which is generally one of the most important predictors of price.**
  Bedrooms & Bathrooms: The number of rooms in a house significantly affects its value. Homes with more rooms tend to be priced higher.
- **Floors: The number of floors in a house could indicate a larger, more luxurious home, potentially raising its price.**
- **Year Built: The age of the house can affect its condition and value. Newly built houses are generally more expensive than older ones.**
- **Location: Houses in desirable locations such as downtown or urban areas tend to be priced higher than those in suburban or rural areas.**
- **Condition: The current condition of the house is critical, as well-maintained houses (in 'Excellent' or 'Good' condition) will attract higher prices compared to houses in 'Fair' or 'Poor' condition.**
- **Garage: Availability of a garage can increase the price due to added convenience and space.**
- **Price: The target variable, representing the sale price of the house, used to train machine learning models to predict house prices based on the other feature.**

FEATURE DISTRIBUTIONS:

- **Area Distribution: The area of the houses in the dataset ranges from 500 to 5000 square feet, which allows analysis across different types of homes, from smaller apartments to larger luxury houses.**

- **Bedrooms and Bathrooms: The number of bedrooms varies from 1 to 5, and bathrooms from 1 to 4. This variance enables analysis of homes with different sizes and layouts.**

- **Floors: Houses in the dataset have between 1 and 3 floors. This feature could be useful for identifying the influence of multi-level homes on house prices.**

- **Year Built: The dataset contains houses built from 1900 to 2023, giving a wide range of house ages to analyze the effects of new vs. older construction.**

- **Location: There is a mix of urban, suburban, downtown, and rural locations. Urban and downtown homes may command higher prices due to proximity to amenities.**

- **Condition: Houses are labeled as 'Excellent', 'Good', 'Fair', or 'Poor'. This feature helps model the price differences based on the current state of the house.**

- **Price Distribution: Prices range between $50,000 and $1,000,000, offering a broad spectrum of property values. This range makes the dataset appropriate for predicting a wide variety of housing prices, from affordable homes to luxury properties.**

CORRELATION BETWEEN FEATURES:

- **A key area of interest is the relationship between various features and house price:**
- **Area and Price: Typically, a strong positive correlation is expected between the size of the house (Area) and its price. Larger homes are likely to be more expensive.**
- **Location and Price: Location is another major factor. Houses in urban or downtown areas may show a higher price on average compared to suburban and rural locations.**
- **Condition and Price: The condition of the house should show a positive correlation with price. Houses in better condition should be priced higher, as they require less maintenance and repair.**
- **Year Built and Price: Newer houses might command a higher price due to better construction standards, modern amenities, and less wear-and-tear, but some older homes in good condition may retain historical value.**
- **Garage and Price: A house with a garage may be more expensive than one without, as it provides extra storage or parking space.**

POTENTIAL USE CASES:

- **The dataset is well-suited for various machine learning and data analysis applications, including:**
- **House Price Prediction: Using regression techniques, this dataset can be used to build a model to predict house prices based on the available features.**
- **Feature Importance Analysis: By using techniques such as feature importance ranking, data scientists can determine which features (e.g., location, area, or condition) have the greatest impact on house prices.**
- **Clustering: Clustering techniques like k-means could help identify patterns in the data, such as grouping houses into segments based on their characteristics (e.g., luxury homes, affordable homes).**
- **Market Segmentation: The dataset can be used to perform segmentation by location, price range, or house type to analyze trends in specific sub-markets, like luxury vs. affordable housing.**
- **Time-Based Analysis: By studying how house prices vary with the year built or the age of the house, analysts can derive insights into the trends of older vs. newer homes.**

SUMMARY OF MISSING VALUES IN THE DATASET:

```
Id            0
Area          0
Bedrooms      0
Bathrooms     0
Floors        0
YearBuilt     0
Location      0
Condition     0
Garage        0
Price         0
dtype: int64
```

PROGRAM CODE:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
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, r2_score
df = pd.read_csv("D:\FS\Case Study\house_price_data.csv")
df.head()
df.info()
df.isnull().sum()
dup = df.duplicated()
print(f"Number of duplicate rows: {dup.sum()}")
# Create a new column to group by decades
df['Decade'] = (df['YearBuilt'] // 10) * 10

# Group by Decade and count the number of buildings (Id)
decade_counts = df.groupby('Decade')['Id'].count().reset_index()

# Create a figure for the plot
plt.figure(figsize=(12, 6))

# Create a barplot showing the number of buildings for each decade
sns.barplot(data=decade_counts, x='Decade', y='Id', palette='Blues_d')

# Add titles and labels
```

```
plt.title('Number of Buildings Built per Decade')
plt.xlabel('Decade')
plt.ylabel('Count of Buildings')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)
plt.grid()

# Show the plot
plt.tight_layout()
plt.show()

condition_counts = df['Condition'].value_counts()

# Create the pie chart
plt.figure(figsize=(6,6))
plt.pie(condition_counts, labels=condition_counts.index, autopct='%1.1f%%',
startangle=90)

# Add a title
plt.title('Distribution of Conditions')

# Display the chart
plt.show()

numerical_columns = ['Area', 'Bedrooms', 'Bathrooms', 'Floors', 'YearBuilt',
'Price']
plt.figure(figsize=(12, 8))
for i, col in enumerate(numerical_columns):
    plt.subplot(2, 3, i + 1)
    sns.boxplot(data=df, y=col)
    plt.title(f'Boxplot of {col}')
plt.tight_layout()
plt.show()


categorical_columns = ['Condition', 'Location', 'Garage']

plt.figure(figsize=(12, 6))
for i, col in enumerate(categorical_columns):
    plt.subplot(1, 3, i + 1)
    sns.countplot(data=df, x=col)
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Condition', y='Price', palette='Blues_d')
plt.title('Price Distribution by Condition')
plt.show()
```

```
label_encoder = LabelEncoder()
for col in categorical_columns:
    df[col + '_encoded'] = label_encoder.fit_transform(df[col])

# Drop the original categorical columns and 'Id' column after encoding
a = df.drop(columns=categorical_columns + ['Id'])  # Combine into a flat list

# Compute the correlation matrix on numeric and encoded columns
corr_matrix = a.corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
X = df[['Area', 'Bedrooms', 'Bathrooms', 'Floors', 'YearBuilt', 'Decade',
        'Condition_encoded', 'Location_encoded', 'Garage_encoded']]
y = df['Price']

# Split the data into 80% training and 20% testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Initialize the Linear Regression model
model = LinearRegression()

# Train the model using the training data
model.fit(X_train, y_train)
# Make predictions on the testing data
y_pred = model.predict(X_test)
# Calculate Mean Squared Error (MSE) and R-squared (R²) score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the evaluation metrics
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R²) score: {r2}')
# Initialize the Random Forest Regression model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model using the training data
rf_model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = rf_model.predict(X_test)

# Calculate Mean Squared Error (MSE) and R-squared (R²) score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
# Print the evaluation metrics
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R²) score: {r2}')
```

LIBRARIES OVERVIEW:

- **pandas: Used for data manipulation and analysis, especially for handling data frames.**
- **numpy: Provides support for numerical operations and handling arrays.**
- **seaborn: A visualization library based on matplotlib, useful for statistical graphics.**
- **matplotlib: A plotting library for creating static, animated, and interactive visualizations.**
- **sklearn: A machine learning library that provides tools for data preprocessing, model selection, and evaluation.**

CODE PROCESS OVERVIEW:

- **Data Loading: Reads house price data from a CSV file.**
- **Data Exploration: Displays initial rows, info, and checks for missing values and duplicates.**
- **Decade Analysis: Creates a new column to group buildings by the decade of construction and plots their counts.**
- **Condition Distribution: Visualizes the distribution of building conditions using a pie chart.**
- **Numerical Analysis: Generates boxplots for key numerical features to check for outliers.**
- **Categorical Analysis: Plots count distributions for categorical variables.**
- **Price vs Condition: Creates a boxplot to show price distribution by building condition.**
- **Encoding Categorical Data: Encodes categorical variables into numerical format.**
- **Correlation Analysis: Computes and visualizes the correlation matrix for numerical and encoded features.**
- **Data Splitting: Divides the dataset into training and testing sets.**
- **Model Training and Evaluation: Trains a Linear Regression model and evaluates it using MSE and R^2 score and Trains a Random Forest Regression model and evaluates it using the same metrics.**

OUTPUT:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2000 entries, 0 to 1999

Data columns (total 10 columns):

 # Column Non-Null Count Dtype

--- ------ -------------- -----

0 Id 2000 non-null int64

1 Area 2000 non-null int64

2 Bedrooms 2000 non-null int64

3 Bathrooms 2000 non-null int64

4 Floors 2000 non-null int64

5 YearBuilt 2000 non-null int64

6 Location 2000 non-null object

7 Condition 2000 non-null object

8 Garage 2000 non-null object

9 Price 2000 non-null int64

dtypes: int64(7), object(3)

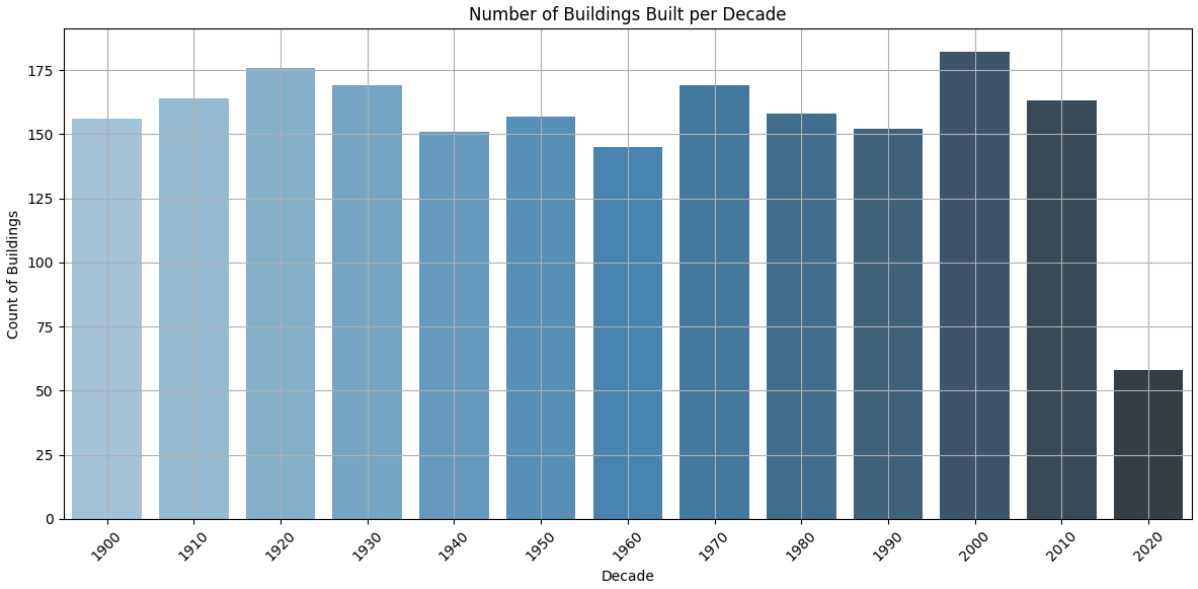
memory usage: 156.4+ KB

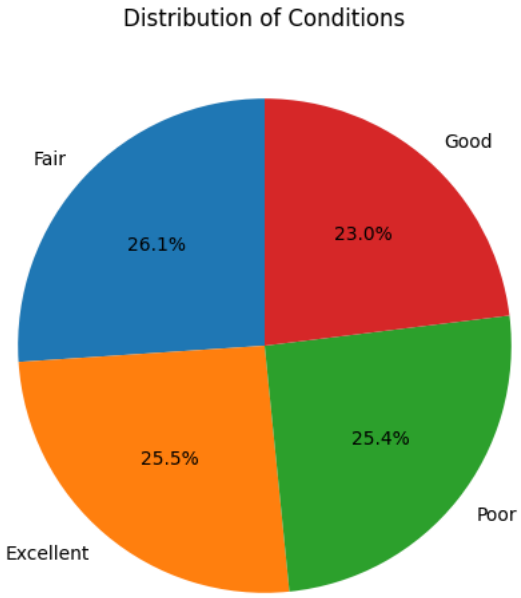
Number of duplicate rows: 0

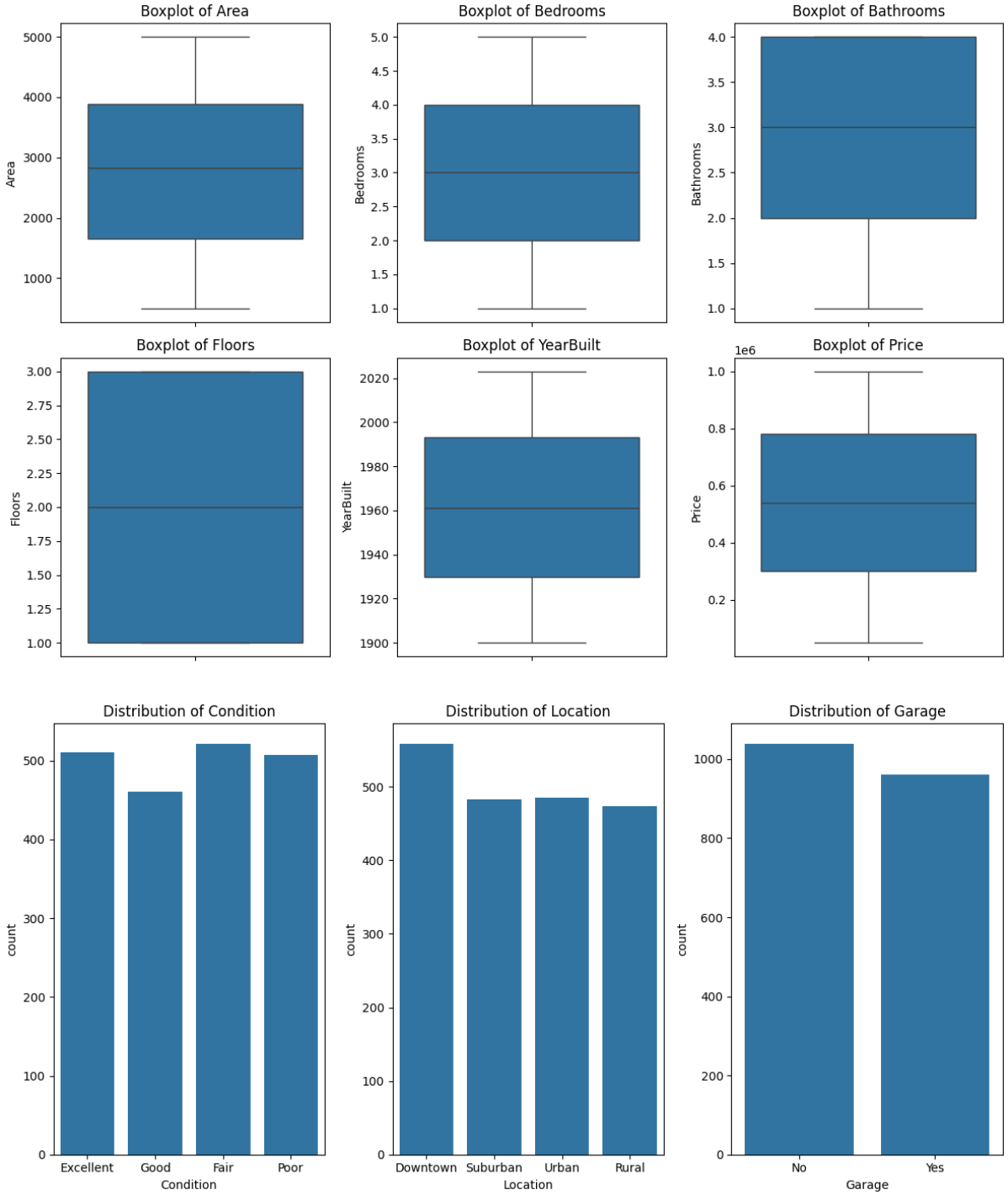
C:\Users\Dell 7400\AppData\Local\Temp\ipykernel_2716\3477118386.py:27: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

sns.barplot(data=decade_counts, x='Decade', y='Id', palette='Blues_d')



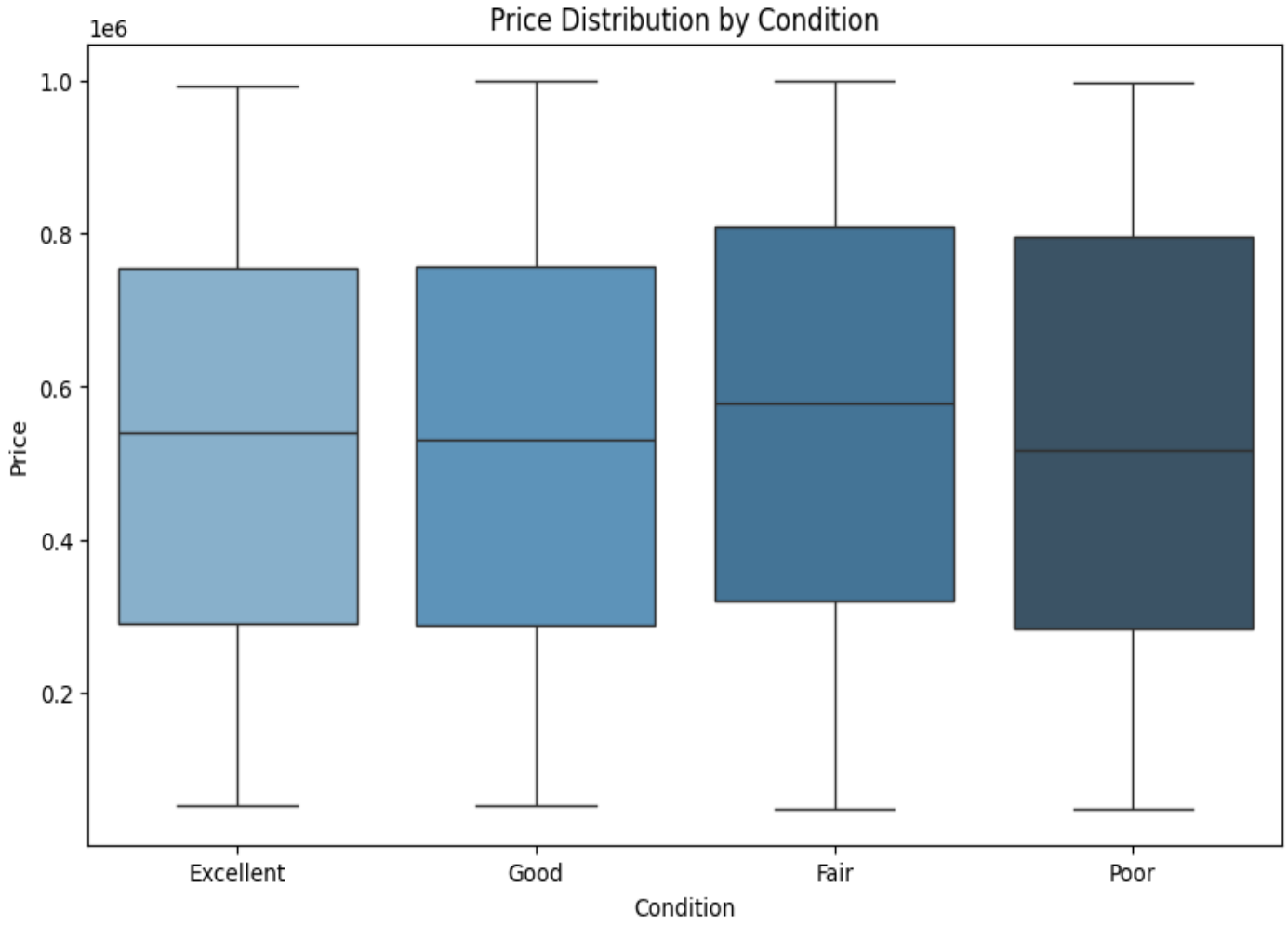


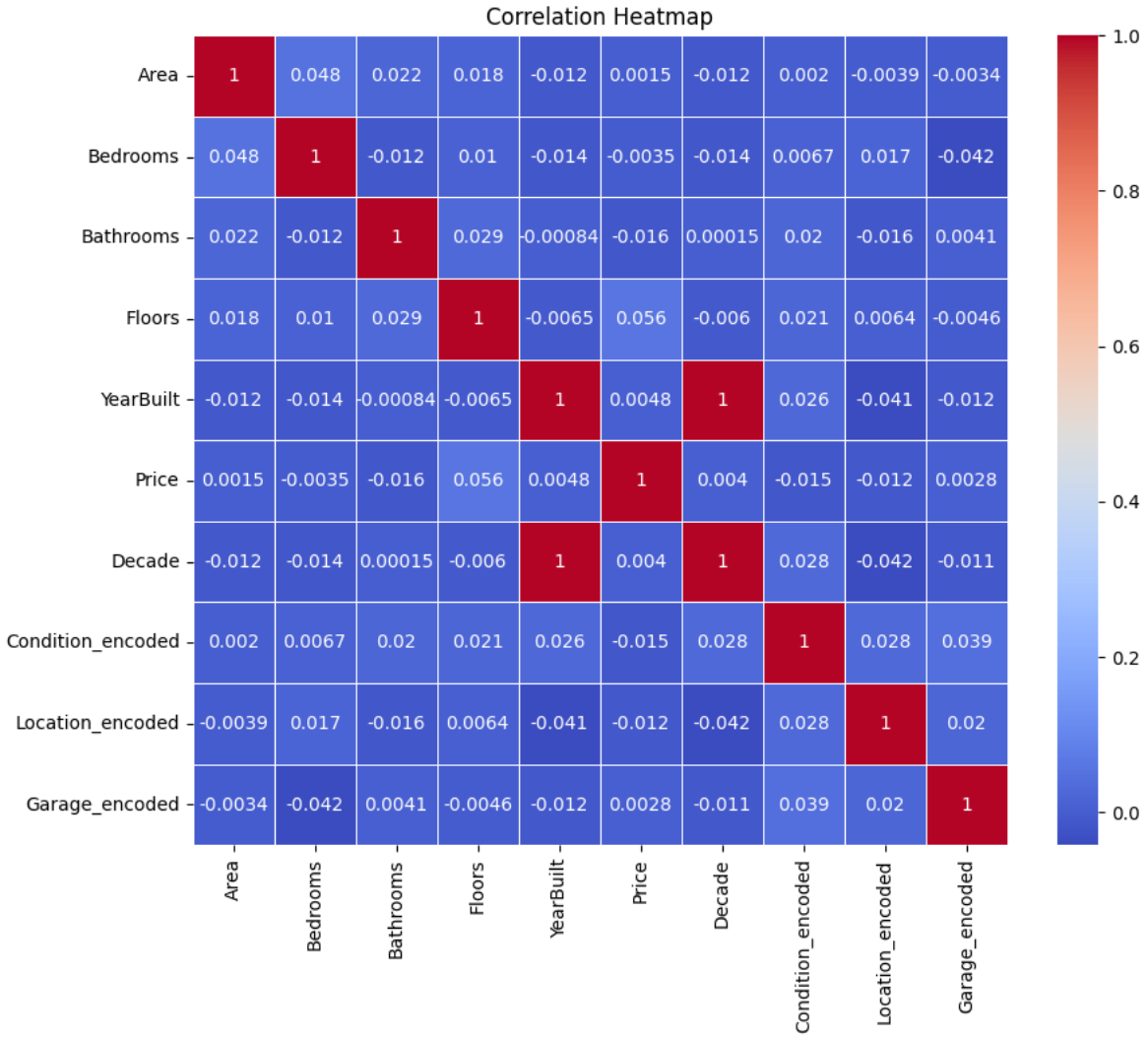


C:\Users\Dell 7400\AppData\Local\Temp\ipykernel_2716\3477118386.py:74: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

sns.boxplot(data=df, x='Condition', y='Price', palette='Blues_d')





Mean Squared Error (MSE): 78922083234.53447

R-squared (R^2) score: -0.014437938668289796

Mean Squared Error (MSE): 86423405283.62074

R-squared (R^2) score: -0.11085741170916696

QUESTION AND ANSWERS:

1 .**What is the purpose of the initial data exploration in the code?**

The initial data exploration aims to understand the dataset's structure, check for missing values, and identify duplicate entries. This step helps to assess the quality of the data before analysis.

2. **How does the code group the data by decades?**

The code creates a new column called `Decade` by integer-dividing the `YearBuilt` column by 10 and then multiplying by 10. This effectively groups the years into decades (e.g., 1990s, 2000s).

3. **What type of visualizations are created to analyze the dataset?**

The code generates a bar chart for the number of buildings built per decade, a pie chart for the distribution of conditions, boxplots for numerical features, count plots for categorical features, and a boxplot showing price distribution by condition.

4 .**Why are categorical variables encoded before modeling?**

Categorical variables are encoded to convert them into numerical format, which is required for most machine learning algorithms to process the data effectively.

5 **.What models are implemented for predicting house prices?**

The code implements two models: Linear Regression and Random Forest Regressor. Each model is trained on the training data, and predictions are made on the testing data.

6. **How is the performance of the models evaluated?**

The performance is evaluated using Mean Squared Error (MSE) and R-squared (R^2) score. These metrics provide insights into the accuracy and goodness of fit of the models.

7. **What improvements are suggested for the code?**

Suggested improvements include handling missing values more effectively, performing hyperparameter tuning for the Random Forest model, using cross-validation for better evaluation, and adding more visualizations to interpret the results.

8. **How can hyperparameter tuning be implemented for the Random Forest model?**

Hyperparameter tuning can be done using `GridSearchCV` to explore different combinations of hyperparameters (like `n_estimators`, `max_depth`, and `min_samples_split`) to find the best performing model.

9. **What does the correlation heatmap visualize?**

The correlation heatmap visualizes the relationships between numeric and encoded variables in the dataset, helping to identify which features are positively or negatively correlated with the target variable (Price).

10**. Why is it important to check for duplicates in the dataset?**

Checking for duplicates is crucial because duplicate entries can skew analysis and lead to biased model predictions, affecting the overall quality of the insights derived from the data.

11**. How does the code handle missing values in the dataset?**

The code checks for missing values using df.isnull().sum(), which returns the count of missing values for each column. However, it does not explicitly handle missing values in the provided code snippet.

12. **What type of plot is used to visualize the number of buildings built per decade?**

A bar plot is used to visualize the number of buildings built per decade, created using sns.barplot.

13**. What type of plot is used to visualize the number of buildings built per decade?**

A bar plot is used to visualize the number of buildings built per decade, created using sns.barplot.

14. **What does the pie chart in the code represent?**

The pie chart represents the distribution of different conditions of the houses in the dataset, showing the proportion of each condition category.