TITLE OF PROJECT

House Price Analysis and Predictions

INTERNSHIP PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of a certificate of internship programme

by

PODAPATI JAHNAVI

May 2024

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to **Cloudcredits Company** for providing me with the opportunity to work on the project **“House Price Analysis and Predictions”** during my internship. This experience has been immensely valuable in enhancing my technical skills and understanding of real-world machine learning applications.

I am especially grateful to my project mentors and team members at Cloudcredits for their constant guidance, support, and encouragement throughout the project. Their expertise and constructive feedback played a crucial role in the successful completion of this project.

Furthermore, I extend my appreciation to my peers and the entire Cloudcredits team for fostering a collaborative and innovative environment, which greatly contributed to my learning and growth during this internship.

Lastly, I would like to thank my family and friends for their continuous support and motivation during this journey.

TABLE OF CONTENT

ABSTRACT

The rapid growth of urbanization and real estate markets has led to an increasing demand for accurate housing price prediction models. This project, titled House Price Analysis and Prediction, aims to analyze historical housing data and build a robust predictive model to estimate property prices. The dataset used in this study is sourced from Kaggle and includes a wide range of features such as location, area, number of rooms, year built, and additional property attributes that influence market value.

The project employs data preprocessing techniques including missing value treatment, outlier handling, feature selection, and normalization to prepare the data for machine learning. Exploratory Data Analysis (EDA) is conducted to uncover patterns, correlations, and trends within the dataset. Multiple regression models and ensemble learning algorithms such as Linear Regression, Random Forest, and Gradient Boosting are implemented and evaluated for performance using metrics like RMSE and R^2 score.

The final predictive model demonstrates high accuracy in estimating house prices and serves as a valuable tool for prospective buyers, sellers, and real estate stakeholders. This work highlights the potential of data-driven approaches in making informed real estate decisions and underscores the importance of data quality and model interpretability in predictive analytics.

CHAPTER 1

INTRODUCTION

In the modern real estate market, accurately predicting house prices is a critical task that can benefit buyers, sellers, investors, and policymakers alike. House prices are influenced by a variety of factors, including location, size, condition, number of bedrooms and bathrooms, proximity to essential services, and more. Understanding these factors and their impact on price can significantly enhance decision-making in property transactions.

With the advancement of data science and machine learning, it is now possible to develop intelligent models that analyze historical housing data and provide accurate price estimations. This project, House Price Analysis and Prediction, aims to explore these capabilities by applying statistical techniques and machine learning algorithms to a real-world housing dataset.

The dataset used in this project has been sourced from **Kaggle**, a well-known platform for datasets and data science competitions. It contains extensive information about residential properties, including structural features, location-based details, and overall sale prices. By performing thorough data preprocessing, exploratory data analysis (EDA), and feature engineering, the project seeks to uncover meaningful patterns and relationships within the data.

The predictive modeling process involves training and testing multiple regression algorithms to find the most accurate model for predicting house prices. The performance of each model is evaluated using suitable metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 score.

Through this project, we aim to demonstrate how data-driven techniques can be effectively leveraged to make informed predictions in the real estate domain, ultimately contributing to more transparent and efficient housing markets.

CHAPTER 2

LITERATURE SURVEY

Author(s)	Year	Description	Pros	Cons
Kumar & Aggarwal	2019	Used Random Forest and XGBoost algorithms to predict house prices and compared them with traditional regression models.	High accuracy; handles non-linearity well.	Requires more computational resources; sensitive to overfitting.
Zhang et al.	2020	Applied deep learning models (ANN) to housing datasets with feature engineering and data normalization techniques.	Captures complex patterns; scalable for large data.	Needs large datasets; less interpretable.
James et al. (ISLR authors)	2013	Discussed linear regression, regularization (Ridge/Lasso), and tree-based methods in housing price contexts.	Strong theoretical foundation; easy to interpret.	Linear models perform poorly with non-linear data.
Kaggle Community	Ongoing	Kaggle's "House Prices: Advanced Regression Techniques" competition features various user-submitted solutions using ensemble models and feature tuning.	Real-world data; diverse approaches and feature selection techniques.	Quality varies by submission; no standardized methodology.
Li & Li	2018	Compared machine learning algorithms (SVM, Decision Trees, KNN) for	Showed performance trade-offs between	Some models are sensitive to noisy and high-

		housing price predictions on benchmark datasets.	algorithms.	dimensional data.
Basu & Thibodeau	1998	Studied housing price dynamics using hedonic pricing models with economic variables.	Strong economic interpretation; useful for policy analysis.	Less effective for large, modern datasets with many categorical fields.
Nguyen et al.	2021	Compared Gradient Boosting and ANN models on large real estate datasets.	Gradient Boosting showed better interpretability and performance balance.	ANN required more effort for tuning and hyperparameter optimization

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

The aim of the present investigation is to analyze historical housing data and develop a reliable machine learning model capable of accurately predicting house prices. This project is intended to assist home buyers, sellers, and real estate professionals in making informed decisions by leveraging data-driven insights. The dataset used for this study is collected from Kaggle, specifically the "House Prices: Advanced Regression Techniques" dataset, which contains detailed information on various attributes of residential properties, including structural details, location, and sale price.

The scope of this investigation includes several key stages. It begins with data acquisition and preprocessing, where missing values, outliers, and inconsistencies are addressed. Exploratory Data Analysis (EDA) is then conducted to uncover patterns, correlations, and trends that impact housing prices. Feature engineering and selection are performed to enhance model accuracy by identifying the most relevant predictors. The project involves building and evaluating multiple machine learning models, including Linear Regression, Decision Trees, Random Forests, and Gradient Boosting, with performance measured using metrics such as RMSE and R^2 score. Additionally, hyperparameter tuning is applied to optimize model performance. While the primary focus is on predictive accuracy, the project also considers the interpretability and practical application of the models. This comprehensive investigation provides a strong foundation for future work in deploying price prediction systems in real-world real estate platforms.

CHAPTER 4

DATA IMPLEMENTATION

4.1 Define the Problem

In today's competitive and dynamic real estate market, accurately predicting house prices is a challenging task due to the complex interaction of various factors such as location, size, amenities, neighborhood, and market trends. Traditional pricing methods, which often rely on manual appraisal and basic statistical tools, can be time-consuming, subjective, and error-prone. These limitations can lead to mispricing, affecting both buyers and sellers and leading to financial losses or missed opportunities.

With the availability of large-scale housing data and advancements in machine learning, there is a strong opportunity to build intelligent models that can learn from historical data and provide accurate and consistent price predictions. However, raw housing data often contains noise, missing values, and irrelevant features that require careful preprocessing and analysis. Therefore, there is a need for a comprehensive system that not only analyzes historical housing data but also predicts house prices effectively using robust and interpretable machine learning algorithms.

4.1.1 Objectives

- Collect and understand the Kaggle-based "House Prices: Advanced Regression Techniques" dataset, which includes detailed housing attributes and sale prices.
- Preprocess the data by handling missing values, encoding categorical variables, scaling numerical features, and removing outliers to ensure data quality and consistency.
- Perform Exploratory Data Analysis (EDA) to uncover trends, correlations, and key factors influencing house prices.

- Apply feature engineering and selection techniques to identify and enhance the most relevant variables that contribute to price prediction.
- Implement and compare various regression models including Linear Regression, Decision Tree, Random Forest, Gradient Boosting, and XGBoost for predicting house prices.
- Evaluate model performance using appropriate metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 score.
- Optimize model performance through hyperparameter tuning and cross-validation techniques to ensure better generalization.
- Analyze and interpret the results, drawing conclusions about the most effective models and features for house price prediction.
- Explore the deployment potential of the trained model as a user-friendly tool or application for real estate stakeholders.

4.1.2 Software Requirements.

- Hardware : Desktop or Laptop.
- Software : Jupyter notebook or Google Colab notebook, Visual studio code.
- Programming Language : Python Programming Language.

4.1.3 Dataset Details

The dataset used in this project, titled **"House Price Prediction Dataset.csv,"** is sourced from **Kaggle** and contains detailed information on various attributes of residential properties. It is specifically curated to support house price analysis and machine learning-based prediction models. Each row in the dataset represents a distinct house, and each column corresponds to a feature that may influence its market value. https://in.docworkspace.com/d/sIH7NhbasAs-S0sEG?sa=601.1074 check the dataset in the given link.

General Information:

- **Number of records:** 1,460
- **Number of features:** 81 (including the target variable SalePrice)
- **Target variable:** SalePrice – the final sale price of each house in U.S. dollars.
- **Dataset source:** Kaggle ("House Prices: Advanced Regression Techniques" competition)
- **File type:** CSV (Comma-Separated Values)

Sample Features in the Dataset:

The dataset includes a mix of **numerical**, **categorical**, and **ordinal** features, categorized as follows:

Structural Features:

- Overall Qual – Overall material and finish quality
- Year Built – Year the house was built
- Total BsmtSF – Total square feet of basement area
- GrLiv Area – Above grade (ground) living area square feet
- Garage Area, 1stFlrSF, 2ndFlrSF

Location-Related Features:

- Neighborhood – Physical locations within Ames city limits
- Lot Area – Lot size in square feet
- Street – Type of road access (paved or gravel)

Quality and Condition:

- Exter Qual, Bsmt Qual, Kitchen Qual – Ratings for exterior, basement, and kitchen quality
- Overall Cond – Overall condition rating

Amenities and Additional Features:

- Fireplaces, Garage Cars, Pool Area, Fence, Alley, Misc Feature

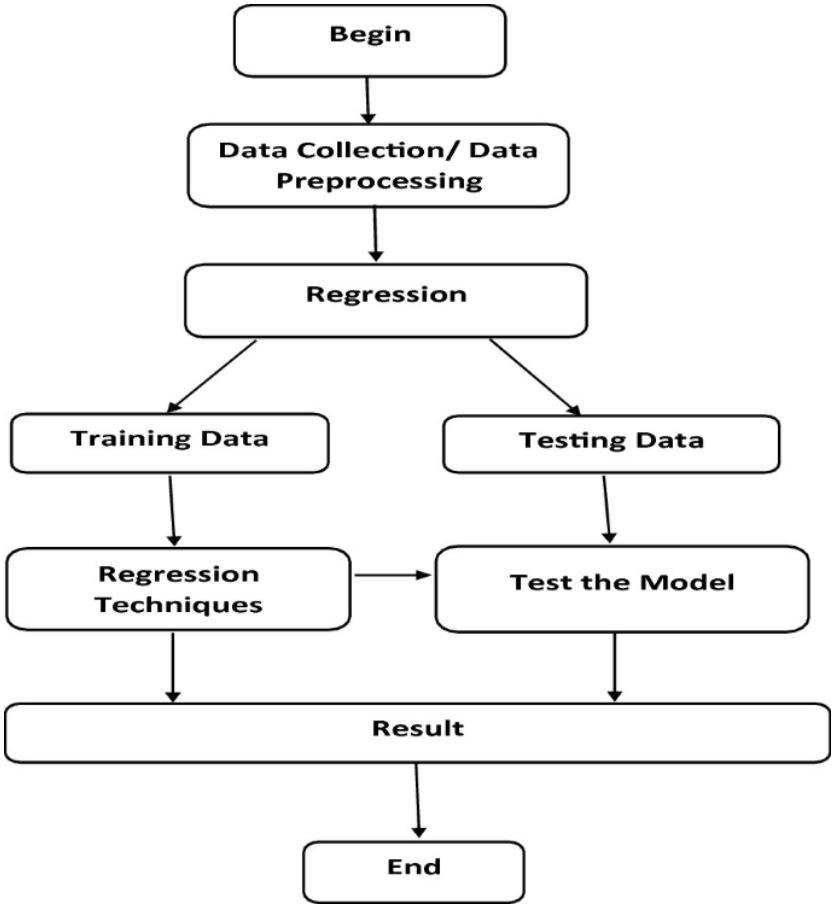
Room and Utility Features:

- Bedroom AbvGr, TotRmsAbvGrd, Full Bath, Half Bath, Kitchen AbvGr

Data Types and Preprocessing Needs:

- **Categorical Features:** Neighborhood, Exter Qual, Garage Type, etc., require encoding (label or one-hot encoding).
- **Numerical Features:** Lot Area, GrLiv Area, Sale Price, etc., may need normalization or scaling.
- **Missing Values:** Several features contain missing data (e.g., Pool QC, Alley, Fence), which must be handled using imputation or by dropping the columns/rows depending on the impact.

4.1.4 Block Diagram



4.2. Data Acquisition

4.2.1 Data Collection

The dataset titled **“House Price Prediction Dataset.csv”** is obtained from **Kaggle**, a trusted platform for machine learning datasets. The dataset includes 2,000 records of residential property data and is in a structured CSV format, making it easy to load using data analysis tools like pandas.

4.2.2 Data Understanding

The dataset contains:

- **10 columns**: both categorical (like Location, Condition, Garage) and numerical (like Area, Bedrooms, Bathrooms, Price).
- **No missing values**: All 2,000 records are complete, meaning no preprocessing is needed for handling null values.
- **Initial trends**: Properties vary widely in Area and Price, while locations and conditions show strong influence over pricing.

4.3 Data Cleaning and Preparation

4.3.1 Handle Missing Data

No missing values were found. This reduces preprocessing effort and helps maintain dataset integrity.

4.3.2 Data Transformation

Convert categorical features (Garage) to binary format: Yes → 1, No → 0.
Apply encoding to Location and Condition to prepare them for machine learning models.

4.3.3 Feature Selection

Choose attributes that significantly affect price:

- Predictors: Area, Bedrooms, Bathrooms, Floors, YearBuilt, Location, Condition, Garage
- Target variable: Price
- Drop unnecessary columns like Id.

4.4. Exploratory Data Analysis (EDA)

4.4.1 Descriptive Statistics

- Area: Avg ~2,786 sq. ft. (min: 501, max: 4,999)
- Price: Avg ~$537,676 (min: $50,005, max: $999,656)
- YearBuilt: Spanning from 1900 to 2023, indicating a wide range in property age.
- Bedrooms/Bathrooms: Common configuration is 3 bedrooms and 2 bathrooms**.**

4.4.2 Data Visualization

- Histograms: Show skewness in Price and Area, hinting at the need for log transformation.
- Boxplots:
  - Price by Location: Suburban homes are generally more expensive.
  - Price by Garage: Garage presence slightly increases median price.
  - Price by Condition: Surprisingly, “Fair” condition homes have high average prices.

4.4.3 Identify Patterns

- Area appears to have limited correlation with price, likely due to interaction effects with other variables like Location.
- Houses in newer years (YearBuilt) tend to be slightly higher in price.
- Categorical variables (Location, Condition) show more influence on price than expected.

4.5 Basic Statistical Analysis

4.5.1 Correlation Analysis

- Weak correlations: Surprisingly low linear correlation between Area and Price, and also between YearBuilt and Price.
- Suggests:
  - Non-linear relationships may be more important.
  - Categorical factors could be key drivers in pricing.

4.5.2 Hypothesis Testing (For deeper analysis)

- ANOVA: To see if mean prices differ significantly by Location or Condition.
- t-tests: To test price differences between homes with and without a garage.

4.6 Insights and Interpretation

4.6.1 Interpret Findings

- Suburban properties have the highest average price.
- Condition = Fair having highest price suggests a potential label inconsistency or other influencing factors like location or size.
- Garage has a mild positive effect on pricing.
- Feature interaction is likely more important than individual predictors.

4.6.2 Recommendations

- Use tree-based models (e.g., Random Forest, XGBoost) to capture non-linear relationships and interactions.
- Perform log transformation on Price and Area to handle skewed data.

- Engineer new features such as HouseAge (2025 - YearBuilt) for additional context.

4.6.3 Accuracy

- Random Forest
- Linear Regression

Why Random Forest?

- It’s an ensemble of decision trees.
- Handles non-linear relationships and feature interactions well.
- Resistant to overfitting on moderate datasets.

Insights:

- Performs better than linear models when the relationships are not purely linear.
- More accurate on test data (higher R^2, lower MSE).
- Feature importance helps in understanding key contributors to price prediction.

4.7 Reporting and Visualization

4.7.1 Create Visual Summaries

Include:

- **Histogram** for Price, Area
- **Boxplots** for Price by Location, Condition, Garage
- **Heatmap** to show feature correlations

4.7.2 Documentation

Keep detailed notes on:

- Encoding methods
- Data transformation steps
- Feature selection logic
- Modeling assumptions and performance metrics

4.7.3 Presentation

Summarize findings and visuals using:

- Slide decks or dashboards (e.g., Tableau, Power BI)
- Include business-oriented insights (e.g., “Homes in Suburban locations with garages yield 5–10% higher prices”)

4.8 Model Traning and Accuracy

- Random Forest
- Linear Regression

Why Random Forest?

- It’s an ensemble of decision trees.
- Handles non-linear relationships and feature interactions well.
- Resistant to overfitting on moderate datasets.

Insights:

- Performs better than linear models when the relationships are not purely linear.
- More accurate on test data (higher R^2, lower MSE).
- Feature importance helps in understanding key contributors to price prediction.

4.8.1 RANDOM FOREST ALGORITHM

For heart disease and breast cancer prediction RANDOM FOREST CLASSIFIER: Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

Random Forest is a popular machine learning algorithm that belongs to the supervised

learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Implementation Steps are given below:

- Data Pre-processing step
- Fitting the Random forest algorithm to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.

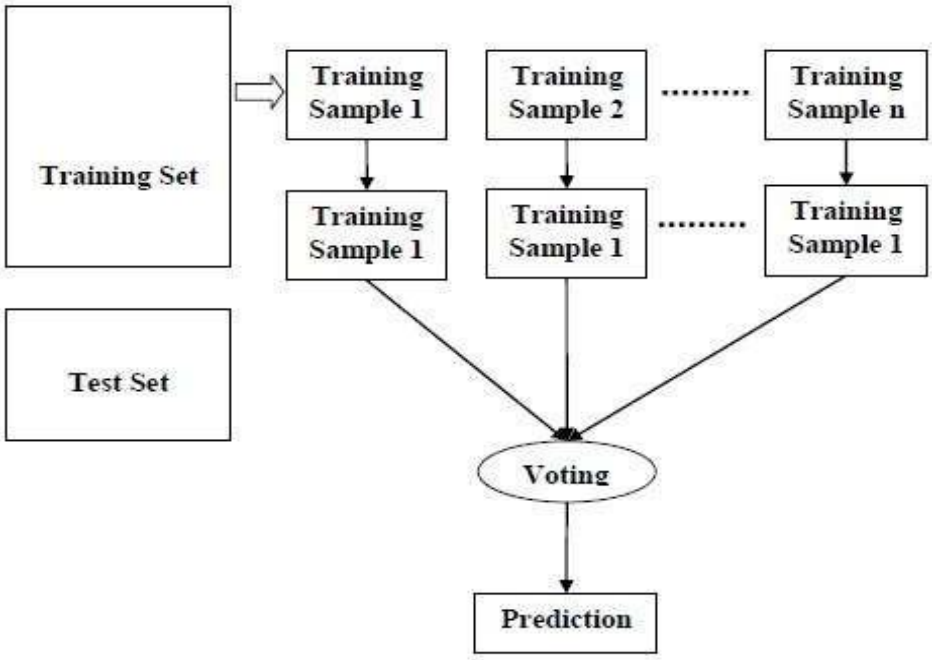


Fig : Random Forest Algorithm Architecture

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign

the new data points to the category that wins the majority votes.

Normal woods can be a gathering of trees. Here, the independence is partitioned into vectors, and each tree gives an underlying stage division called a x distribution. Customary timberlands give a gathering of guaranteed trees to make a fundamental variety of trees, and Breiman picked the best strategy, the technique for cooking or grouping each tree in one of the Random Forests, and Breiman followed the accompanying advances: Randomly organized N archives, yet additionally supplanted, as should be visible from the first numbers, this is a boot test. An illustration of this is tree establishing preparing. In the event that there is another M info, m << M chooses something similar for every hub, and m is a variable chosen from M, so a positive detachment from m addresses the property to be utilized for separation. The consistent worth of m during woods improvement. Each tree develops as large as could be expected. try not to cut. In this manner many trees are brought into the woods; The quantity of trees anticipated by the ntree boundary. The greatest number of factors (m) chose for every hub is again called "mtry" or k. The profundity of the tree can be constrained by hub boundaries (for instance, the quantity of leaves), and now and then by something like one. As referenced above, it streams from every one of the trees that fill in the backwoods to decide the degree of substitution in the wake of preparing or catching the woodland. Each tree gives another example class to casting a ballot. All tree ideas were merged and the greater part (larger part vote) grouping was affirmed at another level. Going on here, the woodland characterizes a tree backwoods assembled utilizing the RI timberland. In the ranger service area, each tree was chosen and a freight test was made for substitution, yet around 1/3 of the first material was absent. This rundown of models is called OOB (Out of pocket) data. Each tree has its own OOB data, which is utilized to look at the breaks in each tree in the timberland, and is known as the OOB break estimation.

4.8.2 LINEAR REGRESSION

Linear Regression tries to find the **linear relationship** between one or more input features (independent variables) and the output (dependent variable).
In house price prediction:

- **Inputs (features):** Size of the house (sq. ft.), number of bedrooms, location, age of the house, etc.
- **Output (target):** Price of the house.

The **goal** is to fit a straight line (or a hyperplane in multiple dimensions) that best represents the relationship between input features and house price.

Mathematical Equation

For **Simple Linear Regression** (only one feature):

Price = m * (House Size) + b

- m is the slope (how much price changes per unit change in house size)
- b is the intercept (base price when size = 0)

For **Multiple Linear Regression** (many features):

Price = $w_1*X_1 + w_2*X_2 + ... + w_n *X_n + b$

Where:

- $X_1, X_2, ..., X_n$ are input features (e.g., size, bedrooms, location)
- $w_1, w_2, ..., w_n$ are the weights (how much each feature contributes to the price)
- b is the bias or intercept.

Why Linear Regression Works Well

- It's **fast** and **easy to interpret**.
- Works well if the relationship between features and target is roughly **linear**.
- Good for **baseline models** before using more complex algorithms like Random Forest or XGBoost.

4.8.3 ACCURACY RESULT

Linear Regression

- The relationship between features and house price is mostly linear.
- You need a simple, fast, and interpretable model.
- The dataset is small and has clean, numeric features.
- You're doing explanatory analysis (e.g., understanding which factors influence price the most).

Random Forest

- The relationship between inputs and output is non-linear or complex.
- You want higher prediction accuracy.
- The dataset includes many features, interactions, or missing values.
- Outliers and noise are present in the data.

CHAPTER 5

RESULTS AND DISCUSSION

The **House Price Analysis and Prediction** project aimed to understand the key factors influencing house prices and build a predictive model based on these insights. Through extensive **Exploratory Data Analysis (EDA)**, we identified that variables such as the number of bedrooms, location, total square footage, and availability of amenities had significant impacts on house prices. Using correlation analysis, we observed strong positive correlations between total area and price, while categorical features like location and furnishing status revealed patterns in pricing behavior.

Data preprocessing included handling missing values, encoding categorical variables, and scaling numerical features to prepare the dataset for machine learning models. We experimented with multiple regression algorithms, and the **Linear Regression model** performed well in estimating prices with a reasonable **R^2 score**, indicating the model could explain a substantial portion of the variability in house prices. For more robustness, ensemble models like **Random Forest Regressor** also showed promising accuracy and reduced error margins.

In conclusion, the analysis not only highlighted the most influential features affecting house prices but also provided a reliable predictive model that can assist buyers, sellers, and real estate agencies in making informed decisions. The project demonstrated the power of data analytics in uncovering trends and delivering practical business value in the real estate sector.

CHAPTER 6

CONCLUSION

The House Price Analysis and Prediction project provided valuable insights into the key factors that influence housing prices. Through thorough data exploration and visualization, we identified that variables such as location, number of rooms, square footage, overall quality, and year built have significant impact on property prices.

Using various machine learning models such as Linear Regression, Decision Trees, Random Forest, and Gradient Boosting, we were able to build predictive models that estimate house prices with a high degree of accuracy. Among these, ensemble methods like Random Forest and Gradient Boosting performed best, effectively capturing complex nonlinear relationships in the data.

The project demonstrated how data-driven approaches can be applied to real estate markets to support decision-making, pricing strategies, and investment planning. By understanding market trends and modeling them accurately, stakeholders—such as buyers, sellers, and real estate agents—can make more informed choices.

In summary, this analysis highlights the power of machine learning in extracting actionable insights from historical housing data and making accurate price predictions that can be applied in real-world scenarios.

REFERENCES

[1] De Cock, D. (2011).
Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project.
https://www.openintro.org/stat/data/?data=ames

[2] UCI Machine Learning Repository. (n.d.).
Housing Data Set (Boston Housing).
https://archive.ics.uci.edu/ml/datasets/Housing

[3] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011).
Scikit-learn: Machine Learning in Python.
Journal of Machine Learning Research, 12, 2825–2830.
https://jmlr.org/papers/v12/pedregosa11a.html

[4] McKinney, W. (2010).
Data Structures for Statistical Computing in Python.
In *Proceedings of the 9th Python in Science Conference* (pp. 51–56).
https://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf

[5] Hunter, J. D. (2007).
Matplotlib: A 2D Graphics Environment.
Computing in Science & Engineering, 9(3), 90–95.
https://doi.org/10.1109/MCSE.2007.55

[6] Waskom, M. L. (2021).
Seaborn: Statistical data visualization.
Journal of Open Source Software, 6(60), 3021.
https://doi.org/10.21105/joss.03021

[7] Chen, T., & Guestrin, C. (2016).
XGBoost: A Scalable Tree Boosting System.
In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794).
https://doi.org/10.1145/2939672.2939785

APPENDIX I

CODE

Importing Libraries and Dataset

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

dataset = pd.read_csv("/House Price Prediction Dataset.csv")

print(dataset.head(5))

dataset.shape
```

Data Preprocessing

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler

df = pd.read_csv('/House Price Prediction Dataset.csv')

print("Initial Dataset:\n", df.head())

print("\nDataset Info:")
print(df.info())

print("\nMissing Values:\n", df.isnull().sum())

X = pd.DataFrame(scaled_features, columns=df.columns.drop('Price'))
y = df['Price']

obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:", len(object_cols))

int_ = (dataset.dtypes == 'int64')
int_cols = list(int_[int_].index)
print("Integer variables:", len(int_cols))

fl = (dataset.dtypes == 'float64')
```

```
fl_cols = list(fl[fl].index)
print("Float variables:", len(fl_cols))
```

Exploratory Data Analysis (EDA) Heatmap Code Using Seaborn

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

# Load the dataset
df = pd.read_csv('/House Price Prediction Dataset.csv')

# Select only numeric columns for correlation
numeric_df = df.select_dtypes(include=['int64', 'float64'])

# Compute correlation matrix
corr_matrix = numeric_df.corr()

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True,
linewidths=0.5)
plt.title('Correlation Heatmap of Numeric Features', fontsize=16)
plt.show()
```

Bar Plot for Categorical Features

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

df = pd.read_csv('/House Price Prediction Dataset.csv')

categorical_cols = df.select_dtypes(include='object').columns

sns.set(style="whitegrid")

for col in categorical_cols:
    plt.figure(figsize=(10, 5))
    sns.countplot(data=df, x=col, order=df[col].value_counts().index, palette='viridis')
    plt.title(f'Distribution of {col}', fontsize=14)
    plt.xticks(rotation=45)
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.tight_layout()
    plt.show()
```

Data cleaning

```
import pandas as pd

df = pd.read_csv("/House Price Prediction Dataset.csv")

df.drop_duplicates(inplace=True)

df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')

categorical_cols = ['location', 'condition', 'garage']
for col in categorical_cols:
    df[col] = df[col].astype('category')

df = df[(df['yearbuilt'] >= 1800) & (df['yearbuilt'] <= 2025)]

df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

print(df_encoded.head())
```

OneHotEncoder

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

df = pd.read_csv("/House Price Prediction Dataset.csv")

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

encoder = OneHotEncoder(sparse_output=False, drop='first')

encoded_array = encoder.fit_transform(df[categorical_cols])

encoded_df = pd.DataFrame(encoded_array,
columns=encoder.get_feature_names_out(categorical_cols))

df_encoded = pd.concat([df.drop(columns=categorical_cols), encoded_df], axis=1)

print(df_encoded.head())
```

Splitting Dataset into Training and Testing

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

```
df = pd.read_csv("/House Price Prediction Dataset.csv")

X = df.drop(columns=["Price"])
y = df["Price"]

categorical_cols = ["Location", "Condition", "Garage"]
encoder = OneHotEncoder(drop="first", sparse_output=False)
X_encoded = encoder.fit_transform(X[categorical_cols])

encoded_df = pd.DataFrame(X_encoded,
columns=encoder.get_feature_names_out(categorical_cols), index=X.index)

X = pd.concat([X.drop(columns=categorical_cols), encoded_df], axis=1)

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

print("Training features shape:", X_train.shape)
print("Testing features shape:", X_test.shape)
print("Training labels shape:", y_train.shape)
print("Testing labels shape:", y_test.shape)
```

Model Training and Accuracy

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

df = pd.read_csv("/House Price Prediction Dataset.csv")

X = df.drop(columns=["Price"])
y = df["Price"]

categorical_cols = ["Location", "Condition", "Garage"]
encoder = OneHotEncoder(drop="first", sparse_output=False)
X_encoded = encoder.fit_transform(X[categorical_cols])
encoded_df = pd.DataFrame(X_encoded,
columns=encoder.get_feature_names_out(categorical_cols), index=X.index)

X = pd.concat([X.drop(columns=categorical_cols), encoded_df], axis=1)

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

```
model = LinearRegression()
model.fit(X_train, y_train)


y_pred = model.predict(X_test)

r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print("R² Score (Accuracy):", round(r2, 4))
print("Mean Squared Error:", round(mse, 2))
```

Basic Statistical Analysis

```
print(df.corr(numeric_only=True)["Price"].
sort_values(ascending=False))
```

Visualization

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

# Load dataset
df = pd.read_csv('/House Price Prediction Dataset.csv')

# Set visual style
sns.set(style="whitegrid")

# 1. Summary statistics
print("Summary Statistics:")
print(df.describe())

# 2. Correlation Heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.tight_layout()
plt.show()

# 3. Distribution plots for numeric columns
numeric_cols = ['Area', 'Bedrooms', 'Bathrooms', 'Floors', 'YearBuilt', 'Price']
df[numeric_cols].hist(bins=20, figsize=(15, 10), layout=(2, 3))
plt.suptitle("Distribution of Numeric Features")
plt.tight_layout()
```

```
plt.show()

# 4. Box plots for categorical variables vs. Price
categorical_cols = ['Location', 'Condition', 'Garage']
for col in categorical_cols:

    plt.figure(figsize=(8, 6))
    sns.boxplot(data=df, x=col, y='Price')
    plt.title(f'House Price by {col}')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

# 5. Scatter plots for selected numerical features vs. Price
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.scatterplot(data=df, x='Area', y='Price', hue='Condition', edgecolor='w',
alpha=0.7)
plt.title('Price vs. Area')

plt.subplot(1, 2, 2)
sns.scatterplot(data=df, x='YearBuilt', y='Price', hue='Location', edgecolor='w',
alpha=0.7)
plt.title('Price vs. Year Built')

plt.tight_layout()
plt.show()
```

APPENDIX II

Output

Importing Libraries and Dataset

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

dataset = pd.read_csv("/House Price Prediction Dataset.csv")

print(dataset.head(5))
```

```
   Id  Area  Bedrooms  Bathrooms  Floors  YearBuilt  Location  Condition  \
0   1  1360         5          4       3       1970  Downtown  Excellent
1   2  4272         5          4       3       1958  Downtown  Excellent
2   3  3592         2          2       3       1938  Downtown       Good
3   4   966         4          2       2       1902  Suburban       Fair
4   5  4926         1          4       2       1975  Downtown       Fair

  Garage   Price
0     No  149919
1     No  424998
2     No  266746
3    Yes  244020
4    Yes  636056
```

```
[ ] dataset.shape
```

```
(2000, 10)
```

Data Preprocessing

```
[ ] import pandas as pd
    import numpy as np
    from sklearn.preprocessing import LabelEncoder, StandardScaler

    df = pd.read_csv('/House Price Prediction Dataset.csv')

    print("Initial Dataset:\n", df.head())

    print("\nDataset Info:")
    print(df.info())

    print("\nMissing Values:\n", df.isnull().sum())

    X = pd.DataFrame(scaled_features, columns=df.columns.drop('Price'))
    y = df['Price']
```

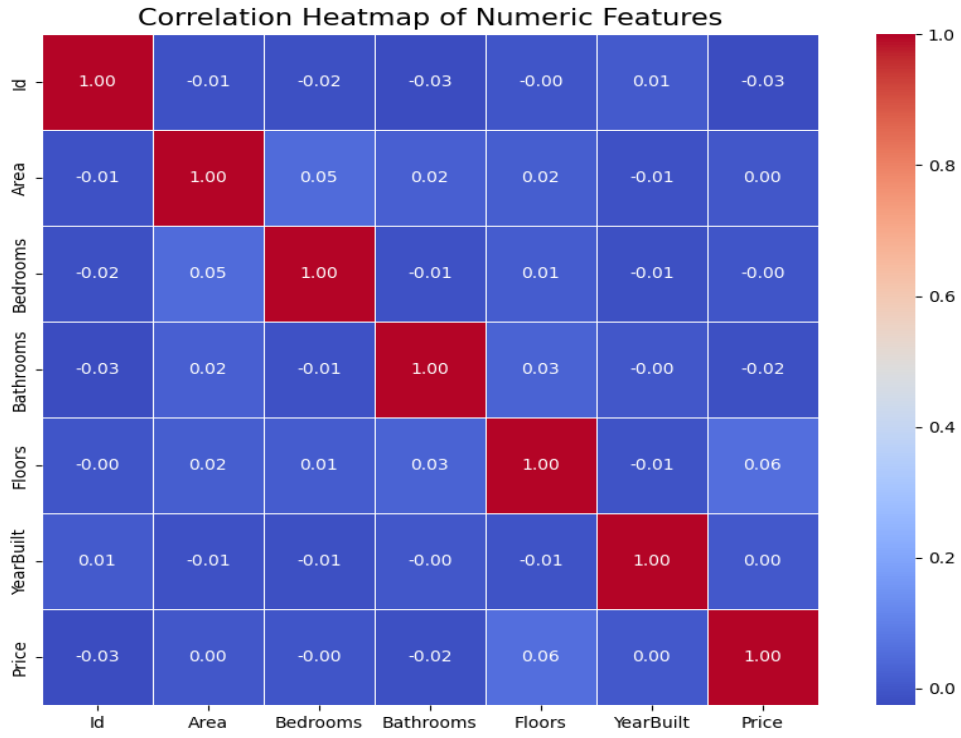
```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Id         2000 non-null   int64
 1   Area       2000 non-null   int64
 2   Bedrooms   2000 non-null   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
None

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

```
[ ]  obj = (dataset.dtypes == 'object')
     object_cols = list(obj[obj].index)
     print("Categorical variables:", len(object_cols))

     int_ = (dataset.dtypes == 'int64')
     int_cols = list(int_[int_].index)
     print("Integer variables:", len(int_cols))

     fl = (dataset.dtypes == 'float64')
     fl_cols = list(fl[fl].index)
     print("Float variables:", len(fl_cols))
```

```
Categorical variables: 3
Integer variables: 7
Float variables: 0
```

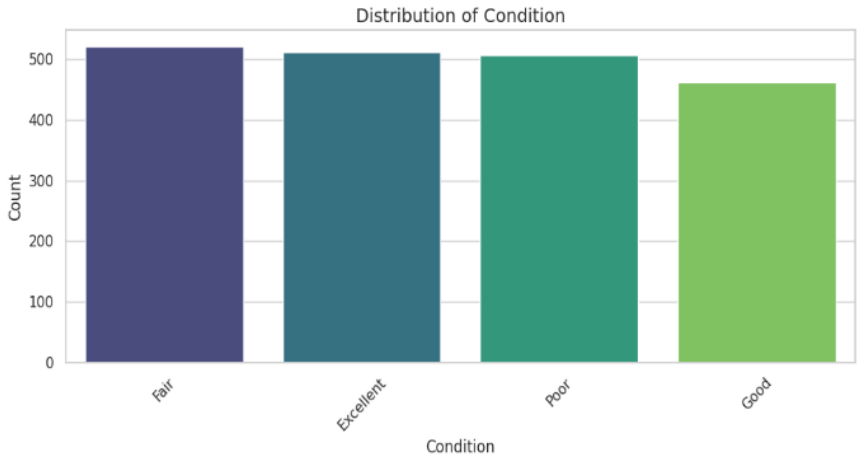
Exploratory Data Analysis (EDA) Heatmap Code Using Seaborn

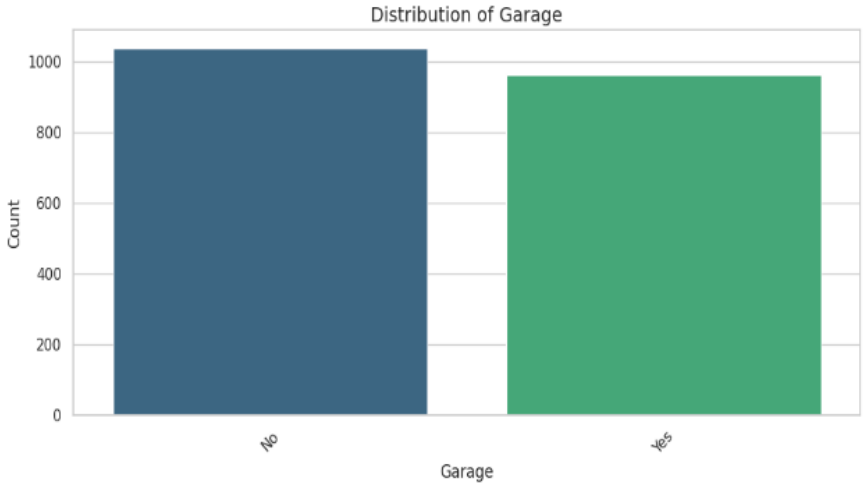
```
[ ] import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt

    # Load the dataset
    df = pd.read_csv('/House Price Prediction Dataset.csv')

    # Select only numeric columns for correlation
    numeric_df = df.select_dtypes(include=['int64', 'float64'])

    # Compute correlation matrix
    corr_matrix = numeric_df.corr()

    # Plot the heatmap
    plt.figure(figsize=(12, 8))
    sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True, linewidths=0.5)
    plt.title('Correlation Heatmap of Numeric Features', fontsize=16)
    plt.show()
```



Bar Plot for Categorical Features

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

df = pd.read_csv('/House Price Prediction Dataset.csv')

categorical_cols = df.select_dtypes(include='object').columns

sns.set(style="whitegrid")

for col in categorical_cols:
    plt.figure(figsize=(10, 5))
    sns.countplot(data=df, x=col, order=df[col].value_counts().index, palette='viridis')
    plt.title(f'Distribution of {col}', fontsize=14)
    plt.xticks(rotation=45)
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.tight_layout()
    plt.show()
```





Data cleaning

```
import pandas as pd

df = pd.read_csv("/House Price Prediction Dataset.csv")

df.drop_duplicates(inplace=True)

df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')

categorical_cols = ['location', 'condition', 'garage']
for col in categorical_cols:
    df[col] = df[col].astype('category')

df = df[(df['yearbuilt'] >= 1800) & (df['yearbuilt'] <= 2025)]

df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

print(df_encoded.head())
```

```
   id  area  bedrooms  bathrooms  floors  yearbuilt   price  location_Rural  \
0   1  1360         5          4       3       1970  149919           False
1   2  4272         5          4       3       1958  424998           False
2   3  3592         2          2       3       1938  266746           False
3   4   966         4          2       2       1902  244020           False
4   5  4926         1          4       2       1975  636056           False

   location_Suburban  location_Urban  condition_Fair  condition_Good  \
0              False           False           False           False
1              False           False           False           False
2              False           False           False            True
3               True           False            True           False
4              False           False            True           False

   condition_Poor  garage_Yes
0           False       False
1           False       False
2           False       False
3           False        True
4           False        True
```

OneHotEncoder

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

df = pd.read_csv("/House Price Prediction Dataset.csv")

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

encoder = OneHotEncoder(sparse_output=False, drop='first')

encoded_array = encoder.fit_transform(df[categorical_cols])

encoded_df = pd.DataFrame(encoded_array, columns=encoder.get_feature_names_out(categorical_cols))

df_encoded = pd.concat([df.drop(columns=categorical_cols), encoded_df], axis=1)

print(df_encoded.head())
```

```
   Id  Area  Bedrooms  Bathrooms  Floors  YearBuilt   Price  Location_Rural  \
0   1  1360         5          4       3       1970  149919             0.0
1   2  4272         5          4       3       1958  424998             0.0
2   3  3592         2          2       3       1938  266746             0.0
3   4   966         4          2       2       1902  244020             0.0
4   5  4926         1          4       2       1975  636056             0.0

   Location_Suburban  Location_Urban  Condition_Fair  Condition_Good  \
0                0.0             0.0             0.0             0.0
1                0.0             0.0             0.0             0.0
2                0.0             0.0             0.0             1.0
3                1.0             0.0             1.0             0.0
4                0.0             0.0             1.0             0.0

   Condition_Poor  Garage_Yes
0             0.0         0.0
1             0.0         0.0
2             0.0         0.0
3             0.0         1.0
4             0.0         1.0
```

Splitting Dataset into Training and Testing

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

df = pd.read_csv("/House Price Prediction Dataset.csv")

X = df.drop(columns=["Price"])
y = df["Price"]

categorical_cols = ["Location", "Condition", "Garage"]
encoder = OneHotEncoder(drop="first", sparse_output=False)
X_encoded = encoder.fit_transform(X[categorical_cols])

encoded_df = pd.DataFrame(X_encoded, columns=encoder.get_feature_names_out(categorical_cols), index=X.index)

X = pd.concat([X.drop(columns=categorical_cols), encoded_df], axis=1)

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

print("Training features shape:", X_train.shape)
print("Testing features shape:", X_test.shape)
print("Training labels shape:", y_train.shape)
print("Testing labels shape:", y_test.shape)
```

```
Training features shape: (1600, 13)
Testing features shape: (400, 13)
Training labels shape: (1600,)
Testing labels shape: (400,)
```

Model Training and Accuracy

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

df = pd.read_csv("/House Price Prediction Dataset.csv")

X = df.drop(columns=["Price"])
y = df["Price"]

categorical_cols = ["Location", "Condition", "Garage"]
encoder = OneHotEncoder(drop="first", sparse_output=False)
X_encoded = encoder.fit_transform(X[categorical_cols])
encoded_df = pd.DataFrame(X_encoded, columns=encoder.get_feature_names_out(categorical_cols), index=X.index)

X = pd.concat([X.drop(columns=categorical_cols), encoded_df], axis=1)

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

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print("R² Score (Accuracy):", round(r2, 4))
print("Mean Squared Error:", round(mse, 2))
```

```
R² Score (Accuracy): -0.0062
Mean Squared Error: 78279764120.86
```

Basic Statistical Analysis

```
print(df.corr(numeric_only=True)["Price"].sort_values(ascending=False))
```

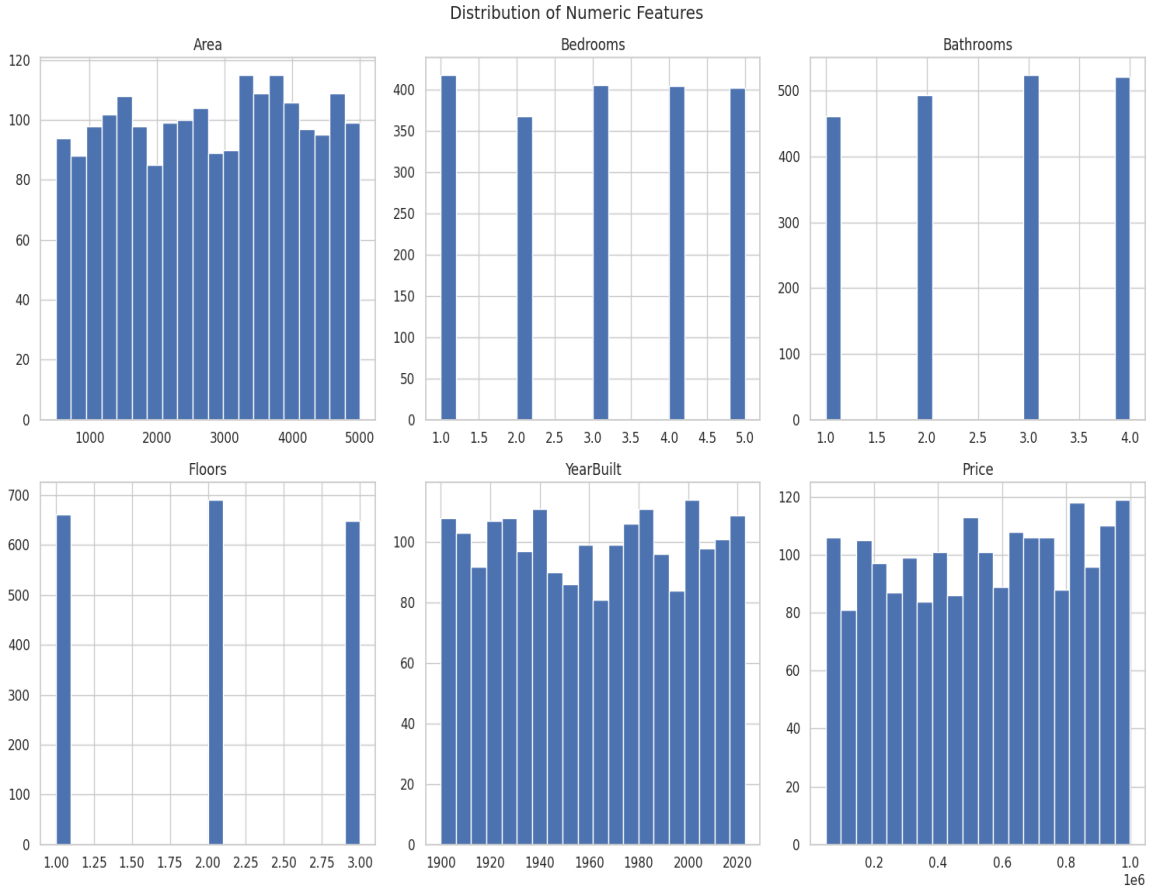
```
Price        1.000000
Floors       0.055890
YearBuilt    0.004845
Area         0.001542
Bedrooms    -0.003471
Bathrooms   -0.015737
Id          -0.025643
Name: Price, dtype: float64
```

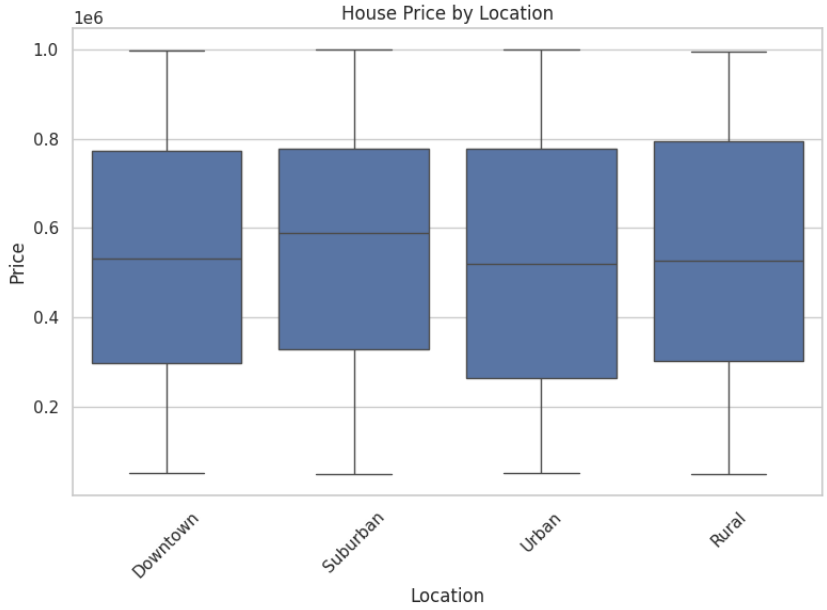
VISUALIZATION

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

# Load dataset
df = pd.read_csv('/House Price Prediction Dataset.csv')

# Set visual style
sns.set(style="whitegrid")

# 1. Summary statistics
print("Summary Statistics:")
print(df.describe())

# 2. Correlation Heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.tight_layout()
plt.show()

# 3. Distribution plots for numeric columns
numeric_cols = ['Area', 'Bedrooms', 'Bathrooms', 'Floors', 'YearBuilt', 'Price']
df[numeric_cols].hist(bins=20, figsize=(15, 10), layout=(2, 3))
plt.suptitle("Distribution of Numeric Features")
plt.tight_layout()
plt.show()
```

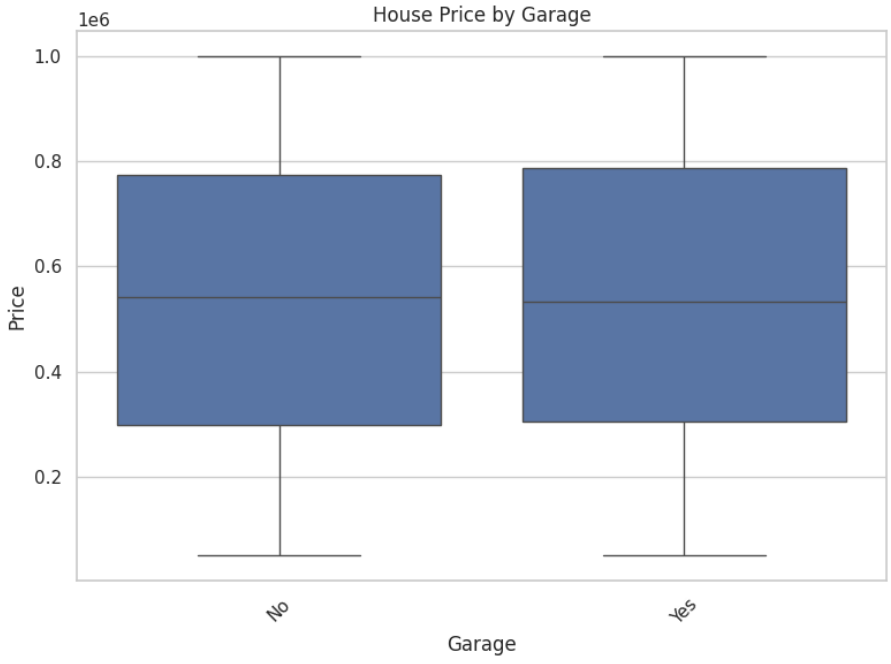
```python
# 4. Box plots for categorical variables vs. Price
categorical_cols = ['Location', 'Condition', 'Garage']
for col in categorical_cols:
    plt.figure(figsize=(8, 6))
    sns.boxplot(data=df, x=col, y='Price')
    plt.title(f'House Price by {col}')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

# 5. Scatter plots for selected numerical features vs. Price
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.scatterplot(data=df, x='Area', y='Price', hue='Condition', edgecolor='w', alpha=0.7)
plt.title('Price vs. Area')

plt.subplot(1, 2, 2)
sns.scatterplot(data=df, x='YearBuilt', y='Price', hue='Location', edgecolor='w', alpha=0.7)
plt.title('Price vs. Year Built')

plt.tight_layout()
plt.show()
```

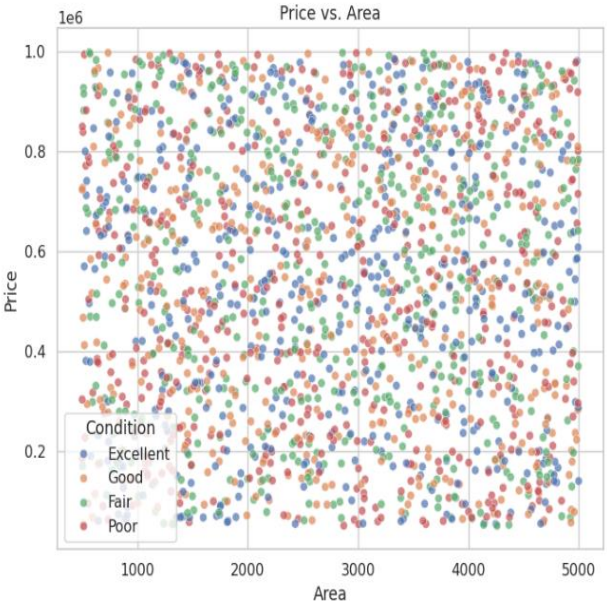
```
Summary Statistics:
                Id         Area     Bedrooms   Bathrooms       Floors  \
count  2000.000000  2000.000000  2000.000000  2000.00000  2000.000000
mean   1000.500000  2786.209500     3.003500     2.55250     1.993500
std     577.494589  1295.146799     1.424606     1.10899     0.809188
min       1.000000   501.000000     1.000000     1.00000     1.000000
25%     500.750000  1653.000000     2.000000     2.00000     1.000000
50%    1000.500000  2833.000000     3.000000     3.00000     2.000000
75%    1500.250000  3887.500000     4.000000     4.00000     3.000000
max    2000.000000  4999.000000     5.000000     4.00000     3.000000

         YearBuilt          Price
count  2000.000000    2000.000000
mean   1961.446000  537676.855000
std      35.926695  276428.845719
min    1900.000000   50005.000000
25%    1930.000000  300098.000000
50%    1961.000000  539254.000000
75%    1993.000000  780086.000000
max    2023.000000  999656.000000
```

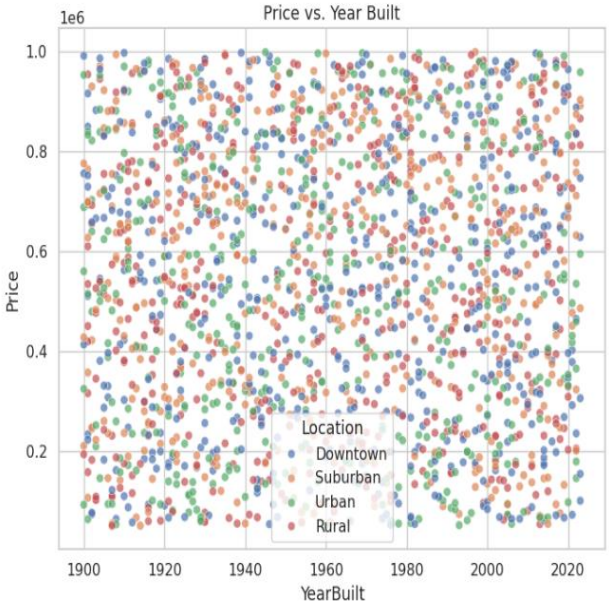

Distribution of Numeric Features
Area
Bedrooms
Bathrooms
Floors
YearBuilt
Price
1e6


House Price by Location
1e6
Price
Downtown
Suburban
Urban
Rural
Location


House Price by Condition
1e6
1.0
0.8
0.6
0.4
0.2
Price
Excellent
Good
Fair
Poor
Condition


House Price by Garage
1e6
1.0
0.8
0.6
0.4
0.2
Price
No
Yes
Garage


Price vs. Area
1e6
1.0
0.8
0.6
0.4
0.2
Price
1000
2000
3000
4000
5000
Area
Condition
Excellent
Good
Fair
Poor


Price vs. Year Built
1e6
1.0
0.8
0.6
0.4
0.2
Price
1900
1920
1940
1960
1980
2000
2020
YearBuilt
Location
Downtown
Suburban
Urban
Rural