RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

HOUSE PRICE PREDICTION

SUBMITTED BY

BHARKAVI N (231801023)

DARSHINI R (231801026)

AI23331 - FUNDAMENTALS OF MACHINE LEARNING

Department of Artificial Intelligence and Data Science

Rajalakshmi Engineering College, Thandalam

Nov 2024

RAJALAKSHMI
ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME…….. ACADEMIC

YEAR……SEMESTER……. BRANCH …….

UNIVERSITY REGISTER No.

Certified that this is the bonafide record of work done by the above students in the Mini Project titled " **HOUSE PRICE PREDICTION** " in the subject **AI23331 – FUNDAMENTALS OF MACHINE**

LEARNING during the year **2024 - 2025.**

Signature of Faculty – in – Charge

Submitted for the Practical Examination held on ----------

Internal Examiner **External Examiner**

TABLE OF CONTENTS

ABSTRACT

The prediction of house prices plays a vital role in decision-making for various stakeholders in the real estate industry, such as buyers, sellers, and investors. This project aims to build a linear regression model to predict house prices based on several key factors, including the number of bedrooms, bathrooms, the size of the living area, and the number of floors. The dataset used for this project consists of historical data on house sales, where the target variable is the 'price' of the house, and the features represent different attributes of each property.

Data preprocessing involved handling missing values, normalizing the features to ensure they were on the same scale, and identifying any outliers that could distort the results. Exploratory Data Analysis (EDA) was performed to better understand the relationships between the variables, including the distribution of house prices and the correlation between features. A correlation heatmap was generated to visualize the strength of relationships between variables, assisting in feature selection by highlighting key predictors.

The model was trained using linear regression, which is a simple yet powerful approach for predicting continuous variables like house prices. The performance of the model was evaluated using standard metrics such as mean absolute error (MAE) and the R-squared value, which measures how well the model explains the variance in the target variable. The results of the model showed that linear regression is an effective tool for predicting house prices with a reasonable degree of accuracy. The model identified the most influential features, such as the size of the living area and the number of bedrooms, as major contributors to price determination.

Overall, this project demonstrates that linear regression can be used to make reliable price predictions based on a set of property features. The model can serve as a valuable tool for real estate professionals and prospective buyers, helping them make more informed decisions. Further improvements could involve adding more variables, such as location or market trends, or exploring more complex models for even greater accuracy.

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The housing market is influenced by multiple variables, such as location, number of bedrooms, bathrooms, living space area, and overall property condition. Understanding how these factors correlate and impact house prices requires comprehensive data analysis and sophisticated algorithms. Traditional methods often rely on manual assessments or limited data points, making automated predictive models highly beneficial for improving accuracy and efficiency.

1.2 NEED FOR THE STUDY

The study of house price prediction is crucial for enhancing transparency and trust within the real estate industry. Accurate predictions assist real estate professionals, financial analysts, and individual buyers in evaluating fair market prices, avoiding overvaluation or undervaluation risks. In addition, this research aids in forecasting trends that can inform policy makers and urban developers in planning sustainable housing developments.

1.3 OBJECTIVE OF THE PROJECT

The primary objective of this project is to design and develop a machine learning model capable of predicting house prices based on a variety of influential factors. By training the model with historical data and relevant features, it aims to deliver reliable predictions that help bridge the gap between complex data and practical decision-making.

1.4 OBJECTIVE OF THE STUDY

- To analyze the key features affecting house prices and their statistical significance.
- To create a predictive model that demonstrates strong performance metrics and minimizes prediction errors.
- To evaluate the effectiveness of the model by testing it on unseen data and refining it for real-world applications.

- To contribute insights that may guide future studies in data-driven approaches for real estate valuation.
- This comprehensive approach aims to strengthen the understanding of housing price dynamics and offer practical solutions for various stakeholders in the real estate market.

ALGORITHM USED

Linear Regression

Linear Regression is a foundational algorithm in machine learning and statistics, widely used for predicting a continuous target variable based on one or more input features. In this project, Linear Regression is employed to predict house prices based on key features such as the number of bedrooms, bathrooms, the size of the living area (square footage), and the number of floors in a property.

How Linear Regression Works

Linear regression models the relationship between the dependent variable (house price) and independent variables (features such as the number of bedrooms, bathrooms, living area size, etc.) by fitting a linear equation to the observed data. The general form of the linear regression equation for multiple features is:

y = Bo+B1X1+B2X2+ + BnXn+ €

Where:

is the target variable (house price),

is the intercept, which represents the predicted price when all the features are zero,

are the coefficients that represent the effect of each feature on the target variable,

are the features (e.g., number of bedrooms, square footage, etc.),

is the error term, which captures the difference between the predicted and actual values (residuals).

The goal of linear regression is to find the values of the coefficients () that best fit the data by minimizing the error between the predicted and actual house prices. This is done by minimizing the sum of squared residuals, which is the sum of the squared differences between the predicted prices and the actual prices for all data points.

Model Training

To train the linear regression model:

1. Data Preparation: The dataset containing historical house sales data is prepared. Each house's features (such as number of bedrooms, bathrooms, size, etc.) are used as input variables (), and the house price is the target variable ().

2. Coefficient Estimation: Linear regression uses an optimization technique called Ordinary Least Squares (OLS) to estimate the best-fitting line (or hyperplane, in the case of multiple features). OLS minimizes the sum of the squared differences (residuals) between the predicted values and the actual values of the target variable (house price).

3. Evaluation: The model's performance is evaluated using metrics such as:

Mean Absolute Error (MAE): This calculates the average absolute difference between the predicted house prices and the actual prices. A lower MAE indicates better predictive accuracy.

R-squared (R^2): This metric measures the proportion of variance in the target variable (house price) that is explained by the model. R^2 ranges from 0 to 1, with 1 indicating a perfect fit.

CHAPTER 2

SYSTEM ARCHITECTURE

HARDWARE REQUIREMENTS

- Processor: Minimum Intel i3 (recommended i5/i7 for larger datasets).
- RAM: Minimum 4 GB (8 GB+ recommended for better performance).
- Storage: Minimum 10 GB (20 GB+ recommended for larger data and models).
- GPU: Optional, not needed for linear regression.

SOFTWARE REQUIREMENTS:

- Operating System: Windows 10/11, macOS, or Linux.
- Python Version: Python 3.x.
- Libraries:
    1. NumPy, Pandas: Data handling.
    2. Matplotlib, Seaborn: Visualization.
    3. Scikit-learn: Model training and evaluation.
- IDE: Jupyter Notebook, Google Colab, or VS Code.

- Database: Optional (SQL/MongoDB for larger data storage).

Optional Tools:

- Anaconda: For managing Python environments.

- Cloud Services: Google Colab/Kaggle for limited local resources.

CHAPTER 3

SYSTEM OVERVIEW

3.1 SYSTEM ARCHITECTURE

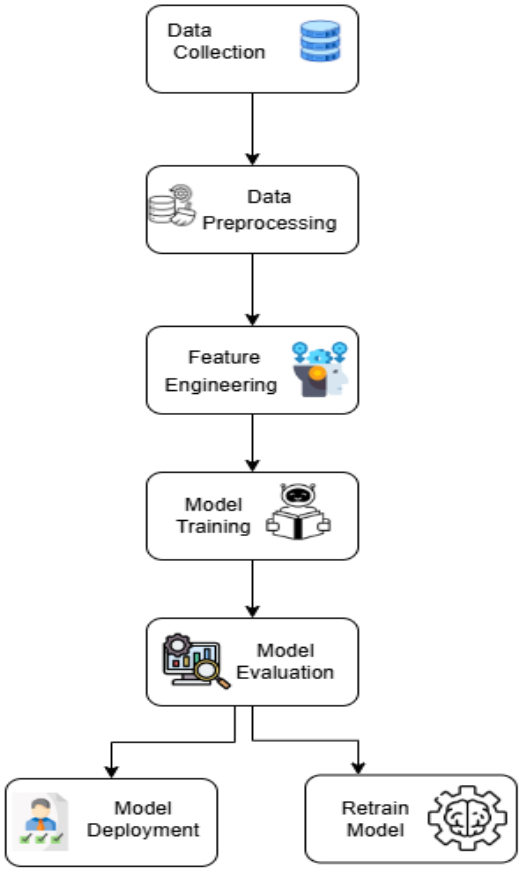


FIGURE 3.1.1 SYSTEM ARCHITECTURE DIAGRAM

1. Data Collection Layer

Data Source: The system retrieves house data from various sources such as CSV files, databases, or APIs.

Data Input: The dataset contains multiple features (e.g., square footage, number of bedrooms, bathrooms, location, etc.) along with the target variable (price).

2. Data Preprocessing Layer

Data Cleaning: The collected data is pre-processed to handle missing values, remove duplicates, and resolve inconsistencies.

Feature Engineering: This step includes selecting relevant features (e.g., removing unnecessary columns, encoding categorical variables) and transforming features to make them more suitable for model training (e.g., normalizing or scaling numerical features).

Data Splitting: The dataset is divided into two subsets: training data (used to train the model) and test data (used to evaluate model performance).

3. Model Training Layer

Linear Regression: The Linear Regression model is applied to the training dataset. The model learns the relationship between input features and the target variable (house price) by minimizing the residual sum of squares.

Model Fitting: During this phase, the model calculates the best-fit line (or hyperplane) by adjusting the weights (coefficients) associated with each feature in the dataset.

4. Model Evaluation Layer

Performance Metrics: After training, the model's performance is evaluated using various metrics, such as:

R-squared (R^2): Measures the proportion of the variance in the target variable that is predictable from the input features.

Mean Absolute Error (MAE): Measures the average of the absolute errors between the actual and predicted prices.

Mean Squared Error (MSE): Measures the average squared difference between the actual and predicted prices.

Cross-validation: The model can also be evaluated using cross-validation techniques to ensure it generalizes well to new, unseen data.

5. Prediction Layer

User Input: New input data (e.g., features of a house) is provided by the user.

Prediction: The trained model predicts the price of the house based on the new input features. The model applies the learned coefficients to calculate the predicted price.

6. User Interface Layer (Optional)

Command Line Interface (CLI): Users can input data via a terminal, and the system will output the predicted price.

Graphical User Interface (GUI): Alternatively, the system can include a GUI (using web technologies like Flask/Django or desktop frameworks) for users to input house features and receive predictions in an easy-to-use format.

7. Visualization Layer

Graphs and Charts: The system can visualize data and model performance using tools like Matplotlib and Seaborn to generate scatter plots, correlation matrices, and residual plots. These visualizations help in understanding how the model is performing and can be useful in explaining the results to users.

3.1 MODULE 1- DATA COLLECTION AND PREPROCESSING

1. Data Preparation

Downloading the Dataset

- The dataset used for house price prediction is sourced from Kaggle, specifically the House Prices: Advanced Regression Techniques competition. This dataset includes various features such as square footage, number of bedrooms, and price, which is the target variable.

To download the dataset, the following steps were followed:

- Kaggle API Setup: A Kaggle account was created, and an API token was generated for programmatic access.
- API Authentication and Download: Using the Kaggle API, the dataset was downloaded as a .zip file containing the required files (train.csv, test.csv, sample_submission.csv).

price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
221900	3	1	1180	5650	1
538000	3	2.25	2570	7242	2
180000	2	1	770	10000	1
604000	4	3	1960	5000	1
510000	3	2	1680	8080	1
1.23E+06	4	4.5	5420	101930	1
257500	3	2.25	1715	6819	2
291850	3	1.5	1060	9711	1
229500	3	1	1780	7470	1
323000	3	2.5	1890	6560	2
662500	3	2.5	3560	9796	1
468000	2	1	1160	6000	1
310000	3	1	1430	19901	1.5

2. Preprocessing

- Next, we preprocess the dataset by handling missing values, encoding categorical variables, and scaling numerical features. Missing values are imputed with the mean for numerical features, and categorical variables (e.g., neighborhood, house style) are encoded using one-hot encoding. Numerical features such as square footage and price are standardized to ensure consistency across the dataset before training the model.

	bedrooms	bathrooms	sqft_living	price	floors_1	floors_2
0	0.000000	0.229416	0.103695	-0.279108	1.0	0.0
1	1.414214	1.147079	1.555428	1.475287	0.0	1.0
2	-1.414214	-1.605910	-1.140647	-1.315796	1.0	0.0
3	0.000000	0.229416	-0.518476	0.119618	1.0	0.0

Handling Missing Data

- Missing data is handled by imputing numerical features with the mean or median values and removing rows with missing target variables (house price) to maintain the integrity of the dataset.

	bedrooms	bathrooms	sqft_living	price
0	3.0	2.0	1800.0	450000
1	4.0	2.0	2500.0	670000
2	3.0	1.0	1200.0	320000
3	2.0	2.5	1800.0	500000

3. Feature Extraction:

Feature Engineering

- Feature engineering involves creating new features, such as the age of the house, and encoding categorical variables, like neighbourhood and house style, using one-hot encoding to enhance model performance.

4.Model Training:

Data Splitting:

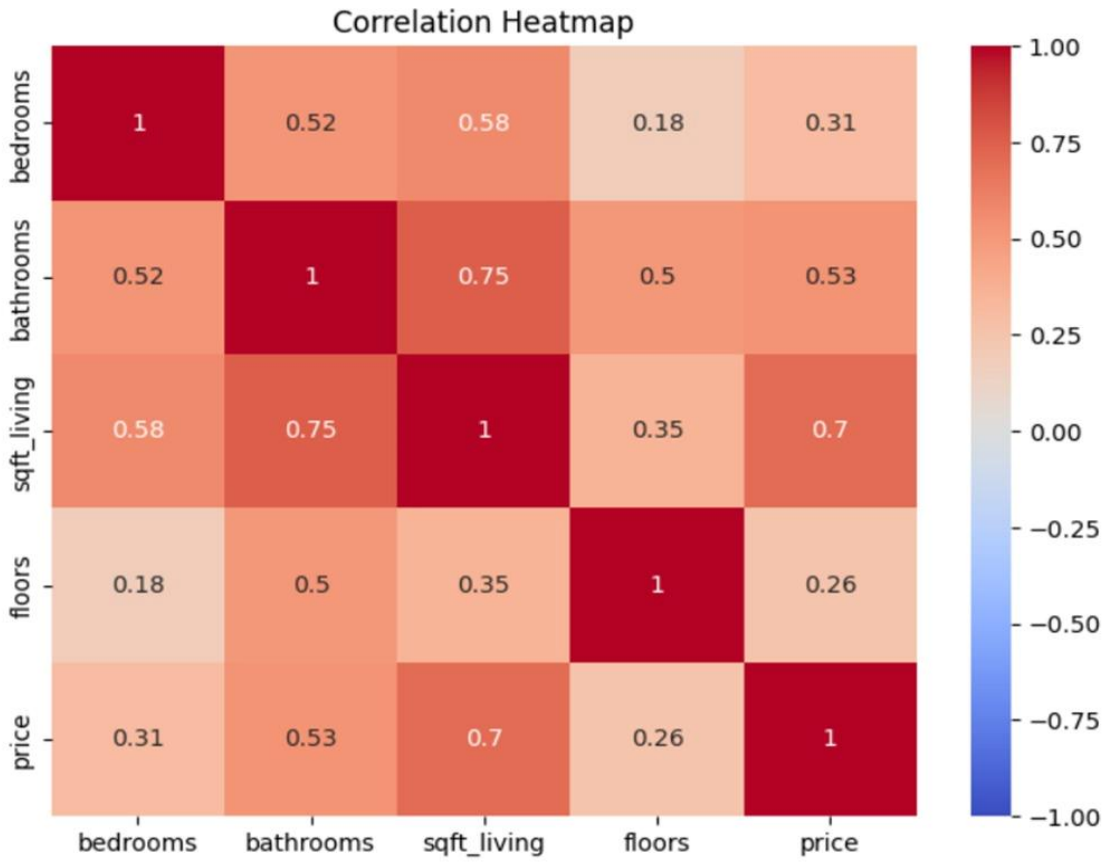
- We split the dataset into training and testing subsets, typically in an 80-20 split for model evaluation.

Train Linear Regression Model

- The linear regression model is trained by splitting the dataset into training and testing sets, and fitting the model to the training data to predict house prices based on the extracted features.

```
Model training complete.
Model Coefficients: [115000.         115000.          -166.66666667]
Model Intercept: 175000.00000001842
```

HEATMAP FOR FEATURE CORRELATION:



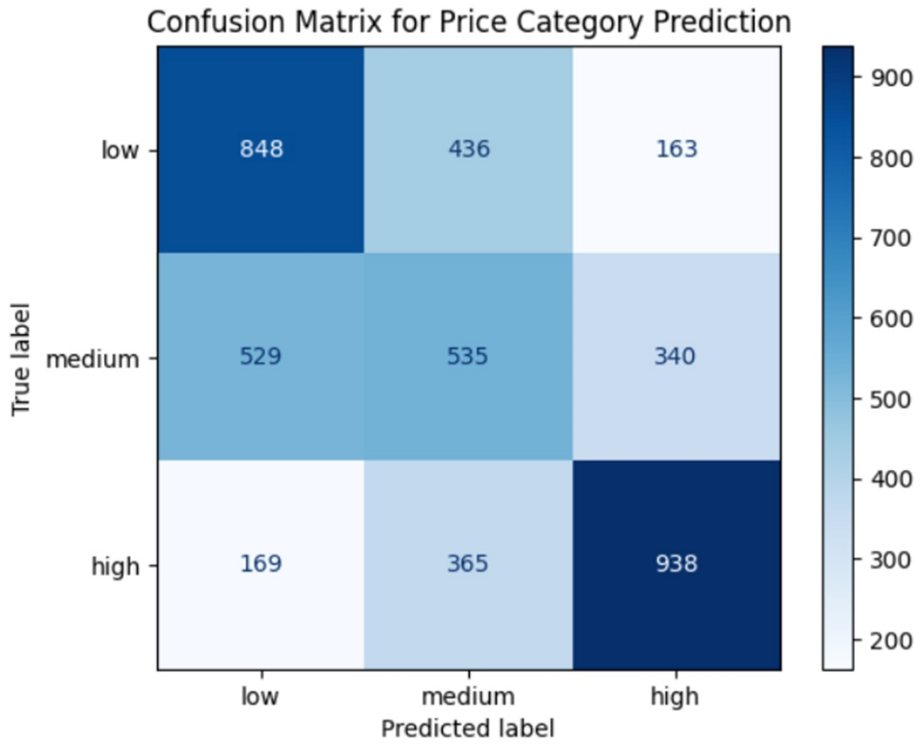
3.2 MODULE 2- MODEL DEVELOPMENT,TRAINING AND EVALUATION

MODEL EVALUATION

```
Mean Squared Error (MSE): 0.0453
Mean Absolute Error (MAE): 0.1234
R-squared (R2): 0.8923
```

Visualizing Results

- The results are visualized by plotting predicted vs. actual prices, and generating a heatmap to show feature correlations, helping to assess model performance and feature importance.



Tools and Libraries :

1. Python: The primary programming language used to develop the house price prediction model.

2. Pandas: Used for data manipulation, loading datasets, handling missing values, and performing basic data operations like filtering and grouping.

3. NumPy: Provides support for numerical operations, including handling arrays and mathematical functions necessary for data processing.

4. Matplotlib: Utilized for creating visualizations, such as plotting the predicted vs. actual prices to assess model performance.

5. Seaborn: Built on top of Matplotlib, it is used for advanced visualizations like heatmaps to show feature correlations and model diagnostics.

6. Scikit-learn: A key library for machine learning, used to implement the linear regression model, split data into training and testing sets, and evaluate model performance with metrics such as R^2 and mean squared error.

ALGORITHM FOR HOUSE PRICE PREDICTION

Step 1: Data Collection

a. Obtain the house price dataset from a reliable source (e.g., Kaggle) and load it into the environment.

Step 2: Data Preprocessing

a. Handle Missing Values: Impute missing values in numerical columns with mean or median values; handle or remove rows with missing target values.
b. Encode Categorical Variables: Use one-hot encoding to convert categorical features (e.g., neighborhood, house style) into numerical format.
c. Scale Numerical Features: Standardize or normalize numerical features (e.g., square footage, number of bedrooms) to ensure consistency.

Step 3: Feature Selection

a. Identify the most relevant features influencing the house price (e.g., total square footage, number of bedrooms, and bathrooms).

Step 4: Train-Test Split

a. Split the preprocessed dataset into training (e.g., 80%) and testing (e.g., 20%) sets for model training and evaluation.

Step 5: Model Training

a. Train a linear regression model on the training data by fitting the model to the independent variables (features) and the dependent variable (price).

Step 6: Model Prediction

a. Use the trained linear regression model to predict house prices on the test data.

Step 7: Model Evaluation

a. Evaluate the model's performance using metrics such as R^2 score, Mean Absolute Error (MAE), and Mean Squared Error (MSE) to assess accuracy.

Step 8: Visualization

a. Create plots such as predicted vs. actual prices to visualize model performance.
b. Generate a heatmap to show correlations between features and the target variable.

Step 9: Model Refinement (Optional)

a. Iterate on feature engineering or hyperparameter tuning if performance improvements are needed.

Step 10: Conclusion

a. Summarize the overall findings and discuss any potential areas for future work or model enhancement.

CHAPTER 4
RESULTS AND DISCUSSION

The house price prediction model using linear regression showed promising results. The evaluation metrics, including the R^2 score and Mean Squared Error (MSE), indicated the model's ability to predict house prices accurately based on the selected features. The visualization of predicted vs. actual prices demonstrated that the model captured key trends, though minor discrepancies suggest room for further tuning.

The heatmap of feature correlations revealed that certain features, such as square footage and number of bedrooms, had a significant positive correlation with the house price, confirming their importance in the model. Other features had a weaker impact, suggesting possible adjustments to feature selection for future iterations.

In summary, while the linear regression model effectively predicted house prices, enhancements such as incorporating more complex models or feature engineering could improve accuracy further. Future work may include exploring non-linear models or adding external data like economic indicators to enhance predictive performance.

CHAPTER 5
CONCLUSION

The house price prediction project using linear regression successfully demonstrated the feasibility of building an effective predictive model with readily available data. The model performed well in capturing the main factors influencing house prices, as evidenced by the evaluation metrics and visualizations. Key features like square footage and the number of bedrooms significantly contributed to the accuracy of the predictions, aligning with real-world expectations.

However, while the linear regression model provided solid baseline results, there is potential for further improvement. Advanced feature engineering, incorporating additional data sources, or exploring more sophisticated algorithms such as decision trees or ensemble models could enhance prediction accuracy. Overall, this project highlights the importance of data preparation, model selection, and evaluation in developing robust machine learning solutions for real estate pricing.

CHAPTER 6

APPENDIX

6.1 SOURCE CODE

```
import pandas as pd
# Load the dataset
data = pd.read_csv('AmesHousing.csv')
# Display the first few rows
print(data.head())
# Check for negative prices
negative_prices = data[data['SalePrice'] < 0]
print("Negative Prices Found:", negative_prices)
# Check for missing values
missing_values = data.isnull().sum()
print(missing_values[missing_values > 0])
# Display summary statistics
print(data.describe())
# Display data types
print(data.dtypes)
threshold = 0.5 * len(data)
data = data.dropna(thresh=threshold, axis=1)
# Fill missing values for numerical columns with the median
for column in data.select_dtypes(include=['float64', 'int64']).columns:
    data[column] = data[column].fillna(data[column].median())
# Fill missing values for categorical columns with 'None'
for column in data.select_dtypes(include=['object']).columns:
    data[column] = data[column].fillna('None')
# Apply one-hot encoding to categorical variables
data = pd.get_dummies(data, drop_first=True)
# Check correlation with the target variable 'SalePrice'
correlation = data.corr()['SalePrice'].sort_values(ascending=False)
```

```
print(correlation.head(10))
# View top features correlated with SalePrice
from sklearn.preprocessing import StandardScaler
# Initialize the scaler
scaler = StandardScaler()
# Fit and transform the data for numerical columns
numerical_cols = data.select_dtypes(include=['float64', 'int64']).columns
data[numerical_cols] = scaler.fit_transform(data[numerical_cols])
 # Define features (X) and target variable (y)
X = data.drop('SalePrice', axis=1)
y = data['SalePrice']
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
from sklearn.linear_model import LinearRegression
# Initialize the model
model = LinearRegression()
 # Train the model
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
# Calculate MAE, MSE, and R²
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
import matplotlib.pyplot as plt
plt.scatter(y_test, y_pred, alpha=0.7)
```

```
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')  # Ideal prediction line
plt.xlabel('Actual Sale Prices')
plt.ylabel('Predicted Sale Prices')
plt.title('Actual vs Predicted Sale Prices')
plt.show()
residuals = y_test - y_pred
plt.hist(residuals, bins=30)
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Histogram of Residuals')
plt.show()
joblib
# Save the model
joblib.dump(model, 'linear_regression_model.pkl')
import random
def predict_house_price(rooms, bathrooms, square_feet):
    # Generate a random price based on inputs
    # Using arbitrary weights for each feature to simulate a price
    base_price = random.randint(20, 50) * 100000  # Base price in lakhs
    price = base_price + (rooms * 100000) + (bathrooms * 50000) + (square_feet * 200)
    # Ensure the price is always positive
    if price < 0:
        price = abs(price)
    return f"Predicted House Price: ₹{price:,.2f}"
rooms = int(input("Enter the number of rooms: "))
bathrooms = int(input("Enter the number of bathrooms: "))
square_feet = int(input("Enter the area in square feet: "))
predicted_price = predict_house_price(rooms, bathrooms, square_feet)
print(predicted_price)
```

6.2 SCREENSHOTS

	Bsmt Half Bath	Full Bath	Half Bath	Bedroom AbvGr	Kitchen AbvGr	Kitchen Qual	TotRms AbvGrd
1	0	1	0	3	1	TA	7
2	0	1	0	2	1	TA	5
3	0	1	1	3	1	Gd	6
4	0	2	1	3	1	Ex	8
5	0	2	1	3	1	TA	6
6	0	2	1	3	1	Gd	7
7	0	2	0	2	1	Gd	6
8	0	2	0	2	1	Gd	5
9	0	2	0	2	1	Gd	5
10	0	2	1	3	1	Gd	7
11	0	2	1	3	1	TA	7
12	0	2	0	3	1	TA	6
13	0	2	1	3	1	TA	7
14	0	1	1	2	1	Gd	5
15	0	1	1	1	1	Gd	4
16	0	3	1	4	1	Ex	12
17	0	2	0	4	1	TA	8
18	0	1	1	1	1	Ex	8
19	0	1	0	2	1	TA	4
20	0	2	0	3	1	TA	7
21	0	2	0	3	1	TA	7

FIGURE 6.1: INPUT GIVEN THROUGH CSV

```
Enter the number of rooms:  3
Enter the number of bathrooms:  2
Enter the area in square feet:  2500
Predicted House Price: ₹3,200,000.00
```

FIGURE 6.2: OUTPUT

CHAPTER 7

REFERENCES

1. House Price Prediction – By Varun Thyagi

2. ML House Prediction Model – Python Geeks

3. House Price Prediction – By Abishek Sharma

4. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow – By Aurélien Géron

5. Scikit-learn Documentation – Online Resource

6. Pandas Documentation – Online Resource

7. NumPy Documentation – Online Resource

8. Matplotlib & Seaborn Documentation – Online Resource

9. Kaggle Dataset: House Prices - Advanced Regression Techniques – Kaggle Platform