PREDICTING HOUSE PRICES WITH SUPERVISED LEARNING

A PROJECT REPORT
for
Introduction to AI (AI101B)
Session (2024-25)

Submitted by

ANTRA PRAKASH
(202410116100035)
AASHI
(202410116100004)
ADITYA SHARMA
(202410116100011)
AYUSH SAINI
(202410116100046)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Mr. Apoorv Jain
Assistant Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206
(APRIL 2025)

TABLE OF CONTENTS

INTRODUCTION

PREDICTING HOUSE PRICES WITH SUPERVISED LEARNING

Housing prices are influenced by numerous factors such as location, income, infrastructure, and demographic details. The ability to predict house prices accurately is of great significance in real estate, banking, and urban planning sectors. Supervised learning, particularly linear regression, provides a straightforward yet powerful technique for such predictive tasks. In this project, we demonstrate the application of a linear regression model to predict house prices using the California Housing datasets.

Predicting housing prices is more than just a data science problem—it addresses real-world issues such as affordability, city planning, investment forecasting, and even economic policy formulation. A well-trained model not only forecasts prices but also uncovers hidden relationships between socioeconomic indicators and housing value.

ASPECTS OF HOUSE PRICE PREDICTION

1. **Economic Relevance**: Predicting house prices supports decisions in investments, tax assessments, and loan approvals.
2. **Data-Driven Decisions**: Real estate developers and agents benefit from understanding the price patterns based on historical data.
3. **Social Impacts**: Accurate predictions help in equitable housing policies and urban planning.
4. **Risk Minimization**: Financial institutions can assess mortgage risk more effectively.
5. **Transparency**: Predictive modeling provides objective and transparent estimations in place of subjective assessments.

PERSPECTIVE

From a machine learning perspective, this problem fits into the domain of regression analysis using supervised learning. We apply a linear regression model, which assumes a linear relationship between the input features and the target variable (median house price). The approach allows interpretation of model coefficients to understand feature importance.

Furthermore, this project provides a foundational understanding of data preprocessing, training and testing split, model evaluation, and visualization. It is a classical case of real-world machine learning deployment.

APPLICATIONS

1. **Real Estate Market Analysis**: Helps buyers and sellers understand fair market value.
2. **Loan Risk Assessment**: Banks use house price prediction models to assess mortgage risk.
3. **Urban Development**: Government bodies use housing trends to allocate resources and plan infrastructure.
4. **Investment Analytics**: Investors leverage price prediction to identify high-growth areas.
5. **Tax Estimation**: Property tax authorities can use predicted values for equitable taxation.
6. **Insurance Underwriting**: Helps insurance providers price their products based on property value trends.

TECHNOLOGICAL BACKGROUND

The technology stack includes:

Python: General-purpose programming language used for scripting and data analysis. Its simplicity and extensive library support make it ideal for machine learning projects. Python is highly readable and supports procedural, object-oriented, and functional programming paradigms.

Pandas and NumPy: Data manipulation and numerical computation libraries. Pandas provides powerful data structures like DataFrames for handling structured data efficiently. NumPy offers support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them.

Scikit-learn: A versatile machine learning library that simplifies the implementation of standard ML algorithms. It provides modules for model selection, preprocessing, evaluation, and deployment. For this project, it facilitates training, testing, and performance evaluation of the regression model.

Matplotlib: A 2D plotting library used to create static, animated, and interactive visualizations. It is essential for examining model performance through plots like scatter diagrams and residual distributions.

Seaborn (optional): Built on top of Matplotlib, Seaborn enhances visualization by adding statistical graphing tools. It is particularly useful for visualizing correlations and distributions during the exploratory data analysis phase.

Jupyter Notebook or IDEs (e.g., VSCode, PyCharm): Provide interactive environments that support code execution, visualization, and documentation. Jupyter Notebook is especially beneficial for step-by-step development and real-time feedback.

METHODOLOGY

1. Data Collection

The project begins with acquiring data essential for building the predictive model.

1.1 **Dataset Source**

The dataset used is the *California Housing dataset* available in the `sklearn.datasets` module. This dataset includes housing data from the 1990 U.S. Census and comprises multiple features relevant to housing prices in California.

1.2 Dataset Features

The dataset includes the following features:

Feature	Description
MedInc	Median income in block group (in tens of thousands)
HouseAge	Median age of houses in the block
AveRooms	Average number of rooms per household
AveBedrms	Average number of bedrooms per household
Population	Total population in the block
AveOccup	Average house occupancy (residents per household)
Latitude	Geographical latitude of the block
Longitude	Geographical longitude of the block
Target (y)	Median house value (in $100,000s)

2. **Data Preprocessing**

Preprocessing ensures the dataset is clean, consistent, and ready for model training.

2.1 **Loading and Cleaning Data**

- The dataset is loaded using `fetch_california_housing()` from scikit-learn and converted into a pandas DataFrame.
- The dataset is checked for null or inconsistent values. Although this specific dataset is clean, real-world datasets often require data imputation.
- Summary statistics and data distributions are reviewed using plots and correlation matrices to understand feature relationships.

2.2 **Feature Engineering**

- Features are selected based on correlation analysis and domain knowledge.
- Although scaling is optional for linear regression, standardization can help with interpretability and visualization.
- Redundant or low-impact features may be dropped to improve model efficiency.

2.3 **Splitting the Dataset**

- The data is split into two sets: 80% for training and 20% for testing.
- This is done using the `train_test_split()` method from scikit-learn, ensuring random but reproducible partitioning.

3.Model Development and Training:

- A Linear Regression model from scikit-learn is used to learn the relationship between input features and the target variable.
- The model is trained using the `fit()` method on the training dataset.
- Model coefficients and intercept are obtained post-training to assess the influence of each feature.

4.Evaluation Metrics

To evaluate how well the model performs on unseen data:

Mean Squared Error (MSE): Measures the average squared difference between predicted and actual values. A lower MSE indicates a better fit.

R^2 Score: Indicates the proportion of variance in the dependent variable that is predictable from the independent variables. Closer to 1 means higher predictive power.

Visualization: A scatter plot of actual vs predicted prices helps visualize model accuracy and identify any visible prediction trends or errors.

CODE

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the California housing dataset
housing = fetch_california_housing()
X = pd.DataFrame(housing.data, columns=housing.feature_names)
y = housing.target

# Split the dataset into training and test sets (80% training, 20% testing)
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 on the training data
model.fit(X_train, y_train)

# Predict house prices on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

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

# Load the California housing dataset
housing = fetch_california_housing()
X = pd.DataFrame(housing.data, columns=housing.feature_names)
y = housing.target

# Split the dataset into training and test sets (80% training, 20% testing)
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 on the training data
model.fit(X_train, y_train)

# Predict house prices on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

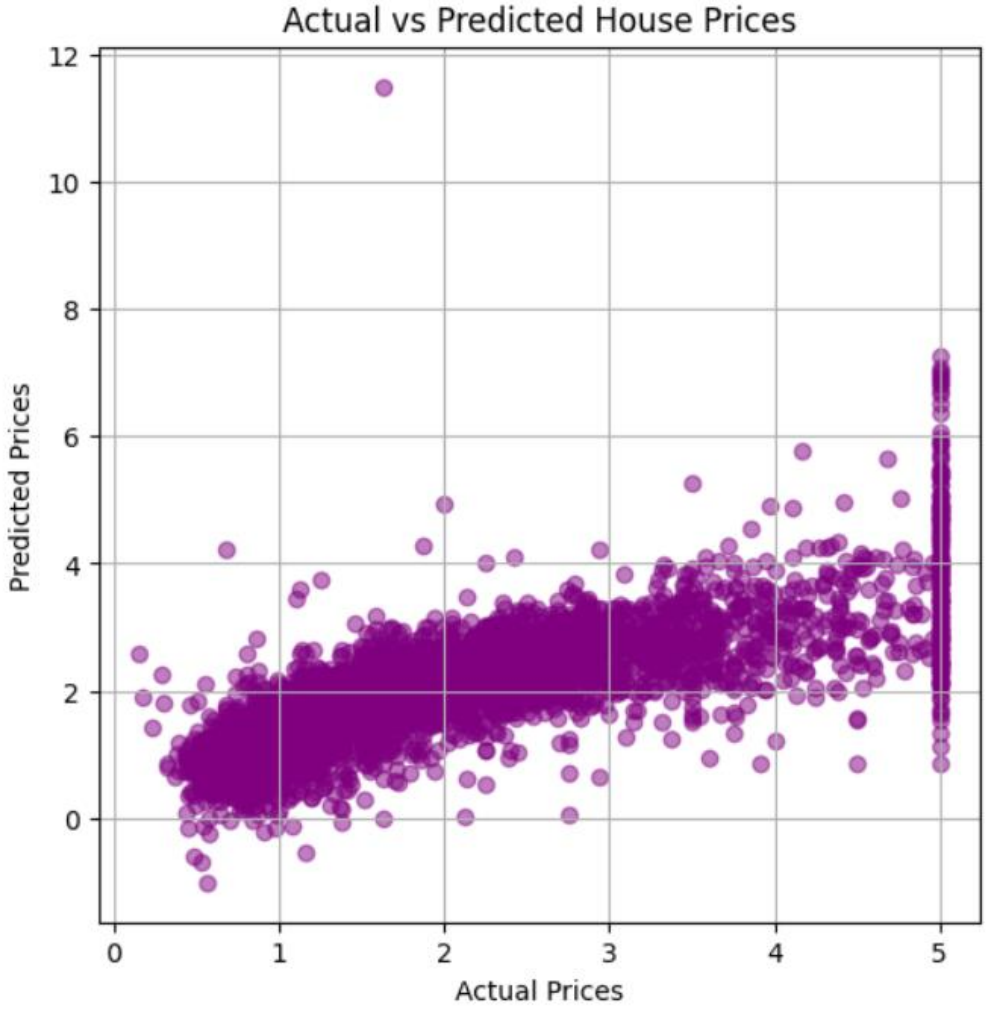
```
print("Mean Squared Error:", mse)
print("R^2 Score:", r2)

# Visualizations

# 1. Actual vs Predicted Scatter Plot
plt.figure(figsize=(6, 6))
plt.scatter(y_test, y_pred, alpha=0.5, color='purple')
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices")
plt.grid(True)
plt.show()

# 2. Histogram of Target Variable
plt.figure(figsize=(8, 5))
plt.hist(y, bins=30, color='skyblue', edgecolor='black')
plt.title("Distribution of House Prices (Target Variable)")
plt.xlabel("Median House Value")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()

# 3. Bar Plot of Feature Coefficients
coefficients = pd.Series(model.coef_, index=X.columns)
plt.figure(figsize=(10, 6))
coefficients.plot(kind='bar', color='teal')
plt.title("Feature Coefficients in Linear Regression Model")
plt.xlabel("Features")
plt.ylabel("Coefficient Value")
plt.grid(axis='y')
plt.show()
```

```
# 4. Residual Plot
residuals = y_test - y_pred
plt.figure(figsize=(8, 5))
plt.scatter(y_pred, residuals, alpha=0.5, color='orange')
plt.axhline(0, color='red', linestyle='--')
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.grid(True)
plt.show()

# 5. Correlation Heatmap of Features
data_with_target = X.copy()
data_with_target['Target'] = y

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

# 6. Line Plot for Predicted vs Actual (Sorted)
sorted_indices = np.argsort(y_test)
sorted_y_test = np.array(y_test)[sorted_indices]
sorted_y_pred = y_pred[sorted_indices]

plt.figure(figsize=(10, 5))
plt.plot(sorted_y_test, label='Actual', color='blue')
plt.plot(sorted_y_pred, label='Predicted', color='green')
plt.title("Actual vs Predicted Prices (Sorted)")
plt.xlabel("Sample Index")
plt.ylabel("House Price")
```

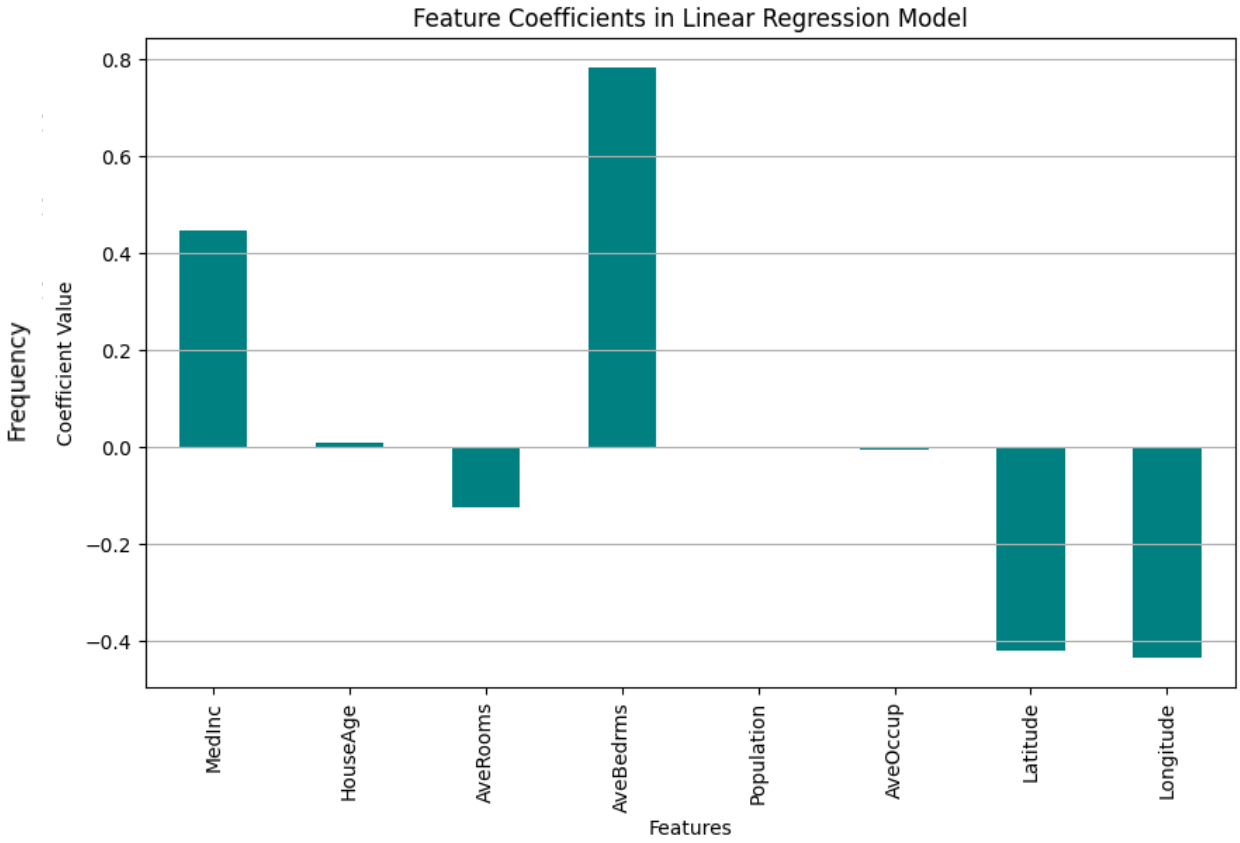
```
plt.legend()
plt.grid(True)
plt.show()
```

OUTPUT

```
Mean Squared Error: 0.5558915986952444
R^2 Score: 0.5757877060324508
```

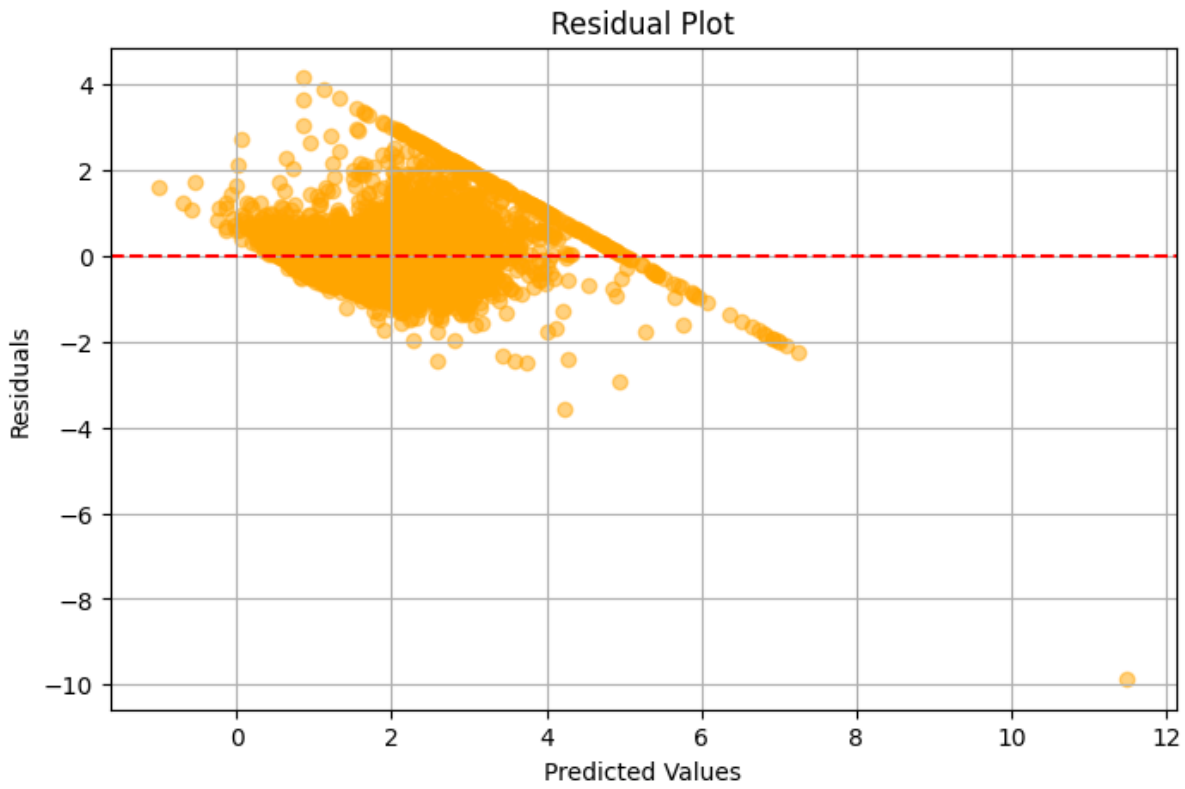


- What it shows: Each dot compares a predicted price (y_pred) to the actual price (y_test).
- Ideal outcome: Dots should lie along the diagonal line (y = x) if predictions are perfect.
- Why it's useful: Helps visualize the accuracy and any consistent over/underestimation by the model.



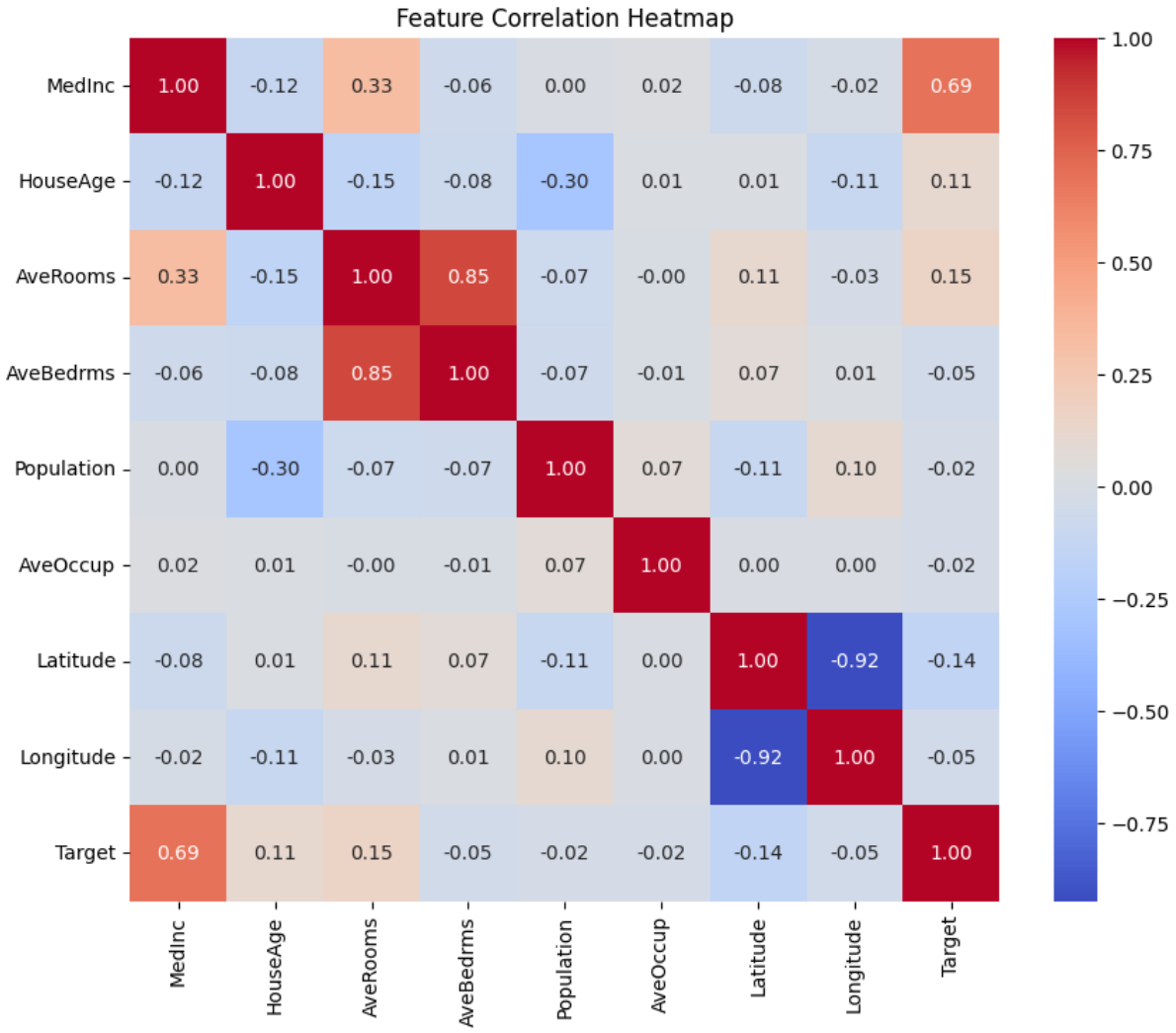
- What it shows: The weight (coefficient) the model assigns to each feature.
- Why it's useful: It tells you which features have the most impact on house price prediction.

  - Positive coefficients → increase the prediction
  - Negative coefficients → decrease the prediction



- What it shows: The difference between actual and predicted values (residuals).
- Why it's useful: You want residuals to be randomly scattered around 0.

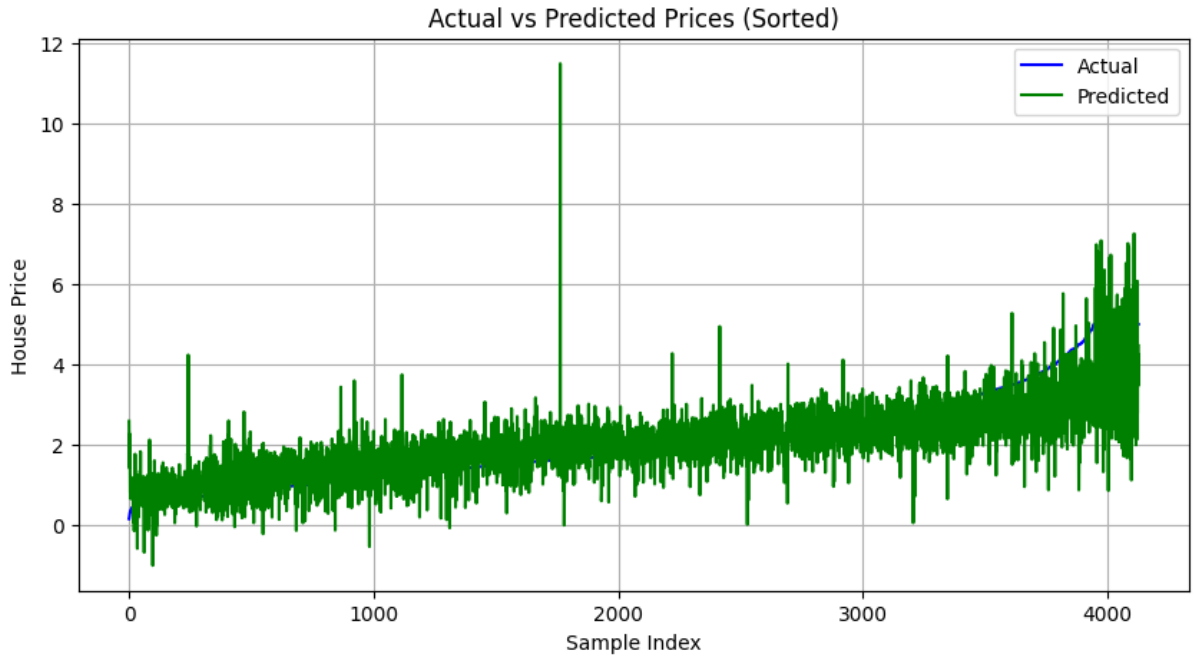
- Patterns (like curves or funnel shapes) indicate model bias or non-linearity that linear regression can’t capture well.



What it shows: Correlation between all features and the target.

Why it's useful:

- High correlation between features and target → useful predictors.
- High correlation between features themselves → multicollinearity, which can affect model interpretation.



What it shows: A side-by-side comparison of predicted vs actual values, sorted by actual price.

Why it's useful:

- If the two lines track each other closely → good prediction.
- Big gaps show where the model is struggling to capture real values.

FUTURE ENHANCEMENTS

1. Model Improvement

- Use advanced regularization techniques like **Ridge Regression** and **Lasso Regression** to reduce overfitting and manage multicollinearity.
- Explore **non-linear models** such as **Random Forests**, **Gradient Boosting Machines**, and **XGBoost** which often provide better accuracy by capturing complex patterns in the data.
- Consider **Support Vector Regression (SVR)** and **K-Nearest Neighbors Regression** for smaller or non-parametric datasets.

2. Feature Engineering

- Introduce location-based features like **proximity to schools, hospitals, transport hubs**, and **shopping centers**.
- Incorporate **environmental attributes** like **air quality, crime rates, and noise levels** which significantly influence property prices.
- Use **interaction terms** and **polynomial features** to model more complex relationships between variables.

3. Real-Time Systems

- Develop APIs that connect the predictive model with real-time property listing platforms.
- Create **interactive web dashboards** where users can input property details and instantly get predictions.
- Integrate predictive models into **CRM systems** used by real estate agencies for personalized client services.

4. Cross-Validation Techniques

- Implement **K-Fold Cross-Validation** for better generalization and to reduce variance between training and testing results.
- Use **Grid Search or Randomized Search** in combination with cross-validation for optimal hyperparameter tuning.

5. Deep Learning Integration

- Utilize **neural networks** such as **MLPRegressor** or **Keras-based models** to capture non-linear relationships in larger, richer datasets.
- Apply **Convolutional Neural Networks (CNNs)** if integrating image data (e.g., satellite images, property images) into the prediction pipeline.

6. Geo-Spatial Analysis

- Use **clustering techniques** like DBSCAN or K-Means to group regions with similar price behavior.
- Implement **spatial regression** models or **GeoPandas** and **Folium** libraries for visual mapping and spatial insights.

7. Temporal Modeling

- Incorporate **historical pricing trends** using **time-series forecasting** models like ARIMA or LSTM.
- Predict future prices by factoring in **seasonality**, **economic indicators**, or **real estate cycles**.

8. Scalable Cloud Deployment

- Deploy trained models on cloud platforms like **AWS SageMaker**, **Google AI Platform**, or **Azure ML Studio.**
- Automate pipelines using **CI/CD** tools and monitor model drift in production environments.

9. Model Explainability and Ethics

- Integrate explainability tools like **SHAP** and **LIME** to communicate model decisions to non-technical users.
- Ensure model predictions are **fair and unbiased**, particularly when used for policy-making or loan evaluations.

10. Educational and Research Use

- Share the project as an **open-source educational toolkit** for learners in machine learning and data science.
- Collaborate with academic institutions to explore **advanced modeling and experimentation**.

CONCLUSION

In this project, we developed a supervised learning model using Linear Regression to predict house prices based on features from the California Housing dataset. Our implementation followed the complete machine learning pipeline—data acquisition, preprocessing, feature analysis, model training, evaluation, and visualization.

Summary of Achievements:

- Successfully loaded and explored the California Housing dataset using Python and Pandas.
- Applied data preprocessing steps to ensure the dataset was clean and analysis-ready.
- Built and trained a Linear Regression model capable of predicting median house values.
- Evaluated model performance using key metrics such as Mean Squared Error (MSE) and R^2 Score.
- Visualized the relationship between actual and predicted prices using scatter plots.
- Gained interpretability into how features like median income and location affect housing prices.

This project proves the feasibility of using simple yet powerful AI models to derive meaningful insights from structured data. It also showcases the potential for extending such models to real-world applications, such as real estate analysis, urban planning, and financial risk assessment.

The foundation laid by this project sets the stage for future enhancements involving more complex models, deployment strategies, and integration with real-time systems. With thoughtful improvements and ethical considerations, this model can evolve into a comprehensive tool for accurate and intelligent property price forecasting.