**Aim -** Build a simple linear regression model to predict house prices based on features like the number of bedrooms and square footage

## Import Statements and Their Purposes

1. import numpy as np

**Purpose**: Provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

2. import pandas as pd

**Purpose**: Used for data manipulation and analysis, including reading data from files like CSV and handling DataFrameoperations.

3. import matplotlib.pyplot as plt

**Purpose**: Essential for creating visualizations such as line charts, scatter plots, and histograms.

4. from sklearn.model_selection import train_test_split

**Purpose**: Provides a method to split the dataset into training and testing subsets, ensuring a controlled and reproducible division.

5. from sklearn.preprocessing import StandardScaler

**Purpose**: Used for feature scaling, ensuring all numerical features have the same scale, typically necessary for machine learning algorithms sensitive to feature magnitudes.

6. from sklearn.linear_model import LinearRegression

**Purpose**: Contains the implementation of linear regression, a simple and widely-used predictive modeling algorithm.

7. import seaborn as sns

**Purpose**: Used for creating visually appealing statistical plots, such as scatter plots with regression lines or heatmaps.

---

## Functions Used and Their Purposes

1. **pd.read_csv()**

○ Reads a CSV file into a Pandas DataFrame.
○ Allows for easy manipulation and analysis of structured data.

2. **dataset.iloc[:, :-1].values**

○ Extracts all columns except the last one (independent variables) as a NumPy array.
○ This isolates the features (X).

3. **dataset.iloc[:, -1].values**

○ Extracts the last column (dependent variable or target variable) as a NumPy array (y).

4. **train_test_split(X, y, test_size=0.2, random_state=42)**

○ Splits the dataset into training (80%) and testing (20%) sets.

○ The random_state ensures reproducibility of the split.

5. **StandardScaler()**

○ Initializes the standard scaler, which normalizes features by removing the mean and scaling to unit variance.

6. **sc.fit_transform(X_train)**

○ Computes the scaling parameters (mean and standard deviation) from the training set and applies scaling to it.

7. **sc.transform(X_test)**

○ Applies the scaling parameters (computed from training data) to the test data, ensuring consistent scaling.

8. **LinearRegression()**

○ Initializes the linear regression model.

9. **regressor.fit(X_train, y_train)**

○ Trains the linear regression model on the training data (X_train and y_train).

10. **regressor.predict(X_test)**

○ Uses the trained model to predict outcomes for the test data.

11. **plt.figure(figsize=(10, 6))**

○ Initializes a new figure for the plot with specified dimensions.

12. **sns.scatterplot()**

○ Creates a scatter plot to visualize the relationship between the actual and predicted prices.

13. **plt.plot()**

○ Plots a reference line representing perfect prediction (y = x).

14. **plt.title(), plt.xlabel(), plt.ylabel(), plt.grid()**

○ Adds a title, axis labels, and gridlines to the plot for better readability.

15. **sc.transform(test_input)**

○ Scales the new input data using the same parameters as the training data.

16. **regressor.predict(scaled_input)**

○ Predicts the house price for the scaled input.

17. **print()**

○ Outputs the predicted price in a human-readable format.

---

## Why Each Step is Necessary

1. **Data Reading and Preparation**:

○ Essential for loading and isolating the features and target variables for analysis.

2. **Splitting the Dataset**:

○ Ensures the model is evaluated on unseen data, avoiding overfitting.

3. **Feature Scaling**:

○ Brings all features to the same scale, crucial for ensuring the algorithm performs optimally and weights features appropriately.

4. **Training the Model**:

○ Builds the predictive relationship between features (X_train) and the target variable (y_train).

5. **Prediction and Visualization**:

○ Provides insights into the model's performance and allows you to predict new, unseen values.

6. **Single Input Prediction**:

○ Demonstrates how to use the trained model to make predictions for custom data.