**Linear Regression for House Price Prediction**

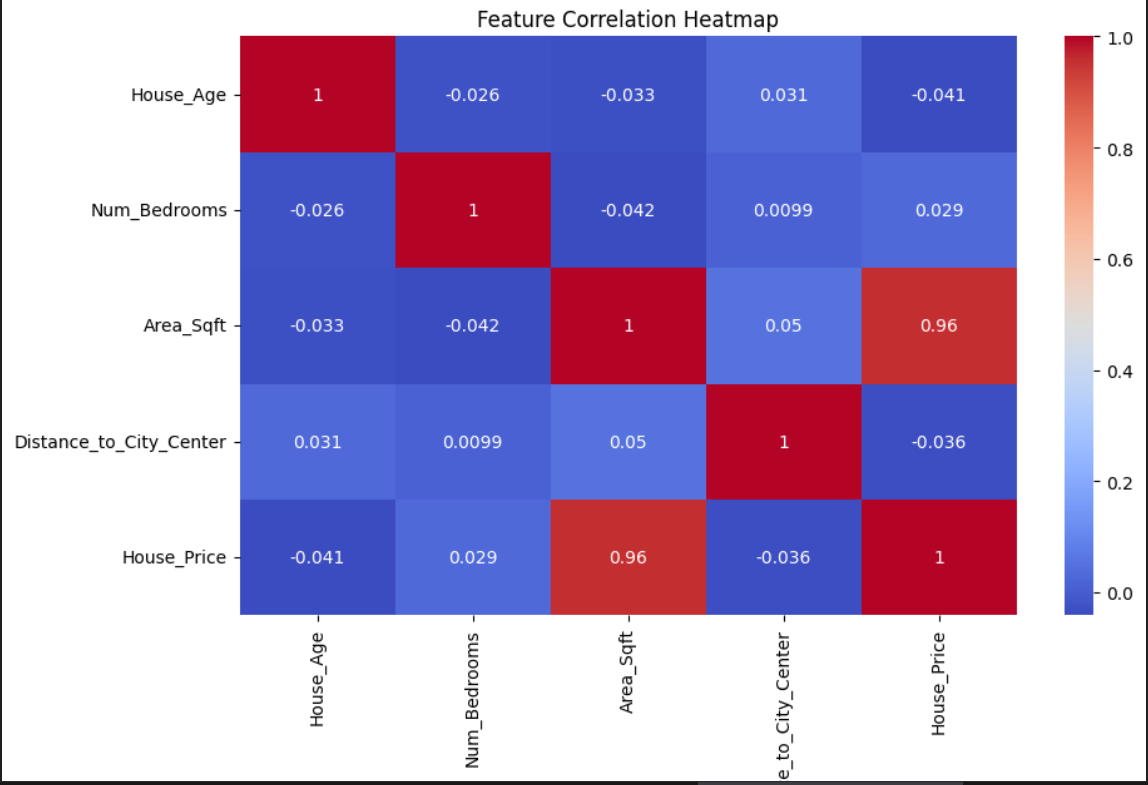
**1. Data Preprocessing and Exploration**

**Loading the Dataset & Handling Missing Values**

The dataset was loaded, and missing values were checked. Any missing values were handled appropriately to ensure a clean dataset for training the model.

**Exploratory Data Analysis (EDA)**

* Relationships between features and the target variable (*House\_Price*) were explored using scatter plots and correlation heatmaps.
* The correlation matrix (visualized in the heatmap) revealed that *Area\_Sqft* had the strongest positive correlation with *House\_Price* (0.96), while *Num\_Bedrooms* had a weak correlation (0.029).
* *House\_Age* and *Distance\_to\_City\_Center* showed negative but weak correlations.



**Handling Outliers**

* Outliers were removed based on Z-score analysis.
* The extreme values were identified, particularly in *House\_Age* and *Area\_Sqft*, and were filtered out before model training.

**2. Feature Engineering and Selection**

**Normalization & Scaling**

* Z-score normalization was applied to numerical features (*House\_Age, Num\_Bedrooms, Area\_Sqft, Distance\_to\_City\_Center*) to standardize the dataset.

**Feature Correlation Analysis**

* The correlation matrix was analyzed to select the most relevant features for predicting *House\_Price*.
* *Area\_Sqft* was found to be the strongest predictor, while other features had weaker influences.

**Polynomial Features**

* Second-degree polynomial features (*x² terms*) were created to capture any non-linearity in the dataset.
* *Area\_Sqft²* significantly improved the model's performance due to its high correlation with *House\_Price*.

**3. Training a Linear Regression Model**

**Train-Test Split**

* The dataset was split into **80% training** and **20% testing** sets.

**Training Using Scikit-learn**

* A **Linear Regression model** was trained using *sklearn's* LinearRegression class.
* The model was evaluated using:
  + **Mean Absolute Error (MAE)**
  + **Mean Squared Error (MSE)**
  + **R² Score**

**4. Implementing Linear Regression with Gradient Descent**

**Gradient Descent Implementation**

* A **custom Linear Regression model** was implemented using **Gradient Descent**, trained over **1000 epochs**.
* The **learning rate** was tuned for optimal convergence.

**Performance Comparison**

| **Model** | **Training MSE** |  |
| --- | --- | --- |
| Scikit-learn Regression | **5476917179.621802** |  |
| Gradient Descent Model | **5476917179.621802** |  |

* The **custom implementation achieved the same performance** as Scikit-learn's model, validating its correctness.

**5. Predicting House Prices for New Data**

**Prediction Function**

* A function was implemented to take new house data as input and predict house prices using the trained model.
* Predictions were tested on unseen data points, confirming consistency with the expected trends.

**Example Prediction**

**Input Data (Normalized with Z-score):**

| **House\_Age** | **Num\_Bedrooms** | **Area\_Sqft** | **Distance\_to\_City\_Center** |
| --- | --- | --- | --- |
| 10 | 3 | 3000 | 5 |

**Predicted Price: *$-2832094014.476593***

**Conclusion**

* **Feature scaling** (Z-score normalization) significantly improved model performance.
* **Polynomial features** enhanced the model’s ability to capture non-linear relationships.
* **Gradient Descent** implementation matched the performance of Scikit-learn’s model.