House Price Prediction Using Linear Regression

Notebook Link:

> House Price Prediction Notebook

1. Introduction

The goal of this project is to predict house prices based on features such as the size of the house, the number of bedrooms, and the age of the house. A linear regression model was implemented from scratch, without using pre-built machine learning libraries such as Scikit-learn. This report summarizes the steps involved, including data preprocessing, model implementation, training, evaluation, and conclusions.

2. Data Preprocessing

2.1 Handling Missing Values

After renaming the columns, the dataset was inspected for missing values. Missing values were found in some of the features. To handle this, the missing values were replaced with the mean of the respective column:

```
df.fillna(df.mean(), inplace=True)
```

This approach ensured that no data points were discarded while maintaining a reasonable estimate for the missing values.

2.2 Loading and Renaming the Data

The dataset used for this project contained the following columns: `Size (sqft)`, `Bedrooms`, `Age`, and `Price`. To make the column names easier to work with, they were renamed to lowercase (`size`, `bedrooms`, `age`, `price`) using the following code:

```
df.rename(columns={
    'Size (sqft)': 'size',
    'Bedrooms': 'bedrooms',
    'Age': 'age',
    'Price': 'price'
}, inplace=True)
```

Renaming the columns ensured consistency in referencing them throughout the code.

2.3 Normalizing the Data

Since the features such as house size, number of bedrooms, and age are on different scales, it was essential to normalize the data to avoid bias in the model. Z-score normalization was applied to scale the features:

```
def normalize_column(column):
    return (column - column.mean()) / column.std()

df['size'] = normalize_column(df['size'])
df['bedrooms'] = normalize_column(df['bedrooms'])
df['age'] = normalize_column(df['age'])
```

Normalization helped bring all features to a similar scale, ensuring that none of the features dominated the others due to their range.

3. Model Implementation

3.1 Linear Regression Overview

The linear regression model predicts the house price by learning the relationship between the target variable (house price) and the features (size, bedrooms, age). The model is based on the following equation:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

Where:

- $\hat{y}$ is the predicted price,
- x_1 is the house size, x_2 is the number of bedrooms, and x_3 is the age of the house,
- $\theta_0, \theta_1, \theta_2, \theta_3$ are the model parameters.

3.2 Model Parameter Calculation

The model parameters were derived using the least squares method, with the following normal equation:

$$\theta = (X^T X)^{-1} X^T y$$

Where X is the matrix of features (with an additional column of ones for the intercept), and y is the target variable (house prices).

The following code was used to calculate the model parameters:

```
theta = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
```

3.3 Prediction Function

A function was implemented to predict house prices based on the learned model parameters:

```
def predict(features, theta):
    features = np.insert(features, 0, 1)  # Add intercept term
    return features.dot(theta)
```

4. Model Training

4.1 Splitting the Dataset

The dataset was split into training and testing sets, with 80% of the data used for training and 20% for testing. The `train_test_split` method from Scikit-learn was used for this purpose.

4.2 Training and Mean Squared Error (MSE)

The model was trained on the training data using the normal equation. The Mean Squared Error (MSE) was calculated to evaluate the performance on the training set:

$$\text{MSE} = \frac{1}{n} \sum\_i = 1^n (\hat{y}\_i - y_i)^2$$

The training MSE was computed as follows:

```
train_mse = np.mean((y_train - y_train_pred) ** 2)
```

5. Model Evaluation

5.1 Testing the Model

The trained model was evaluated on the testing set, and the MSE for the test data was calculated. A lower MSE indicates that the model generalizes well to unseen data. The test MSE was calculated similarly to the training MSE.

5.2 Visualizing the Results

To visualize the model's performance, the predicted house prices were plotted against the actual house prices for the test data. This allowed us to assess how well the model fits the data.

```
plt.scatter(X_test[:, 1], y_test, color='blue', label='Actual')
plt.scatter(X_test[:, 1], y_test_pred, color='red', label='Predicted')
plt.xlabel('Size (Normalized)')
plt.ylabel('Price')
plt.legend()
plt.show()
```

The plot shows a reasonable fit of the regression line, indicating that the model captured the general trend of the data.

6. Challenges Encountered

6.1 Handling Missing Data

One of the initial challenges was dealing with missing values in the dataset. Various strategies, such as dropping rows or filling them with mean values, were considered. The final approach involved replacing missing values with the column mean, which maintained the integrity of the dataset without losing any data points.

6.2 Feature Scaling

Another challenge was ensuring that the features were scaled appropriately. Without normalization, features with larger numerical ranges (e.g., house size) could have dominated the model's predictions. Normalizing the data resolved this issue.

6.3 Renaming Columns

The dataset originally had inconsistent column names (e.g., `Size (sqft)`, `Bedrooms`). Renaming the columns to lowercase provided consistency and made the code easier to read and manage.