Documentations:

About the data Set (HousingPrices-Amsterdam-August-2021)

1. Importing Libraries

Python Code

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline
```

In this section, the necessary libraries for the data analysis are imported, including Pandas, NumPy, Seaborn, and Matplotlib. `%matplotlib inline` is a magic command in Jupyter Notebook to display plots directly in the notebook.

---

2. Loading Data

Python Code

```
df = pd.read_csv("HousingPrices-Amsterdam-August-2021.csv")
```

	Unnamed: 0	Address	Zip	Price	Area	Room	Lon	Lat
0	1	Blasiusstraat 8 2, Amsterdam	1091 CR	685000.0	64	3	4.907736	52.356157
1	2	Kromme Leimuidenstraat 13 H, Amsterdam	1059 EL	475000.0	60	3	4.850476	52.348586
2	3	Zaaiersweg 11 A, Amsterdam	1097 SM	850000.0	109	4	4.944774	52.343782
3	4	Tenerifestraat 40, Amsterdam	1060 TH	580000.0	128	6	4.789928	52.343712
4	5	Winterjanpad 21, Amsterdam	1036 KN	720000.0	138	5	4.902503	52.410538

The dataset is loaded into a Pandas DataFrame called `df` from a CSV file.

---

3. Renaming and Dropping Columns

Python Code

```
df.rename(columns={"Unnamed: 0": "index"}, inplace=True)

df.drop(columns={"index", "Address", "Zip"}, inplace=True)
```

The "Unnamed: 0" column is renamed to "index" and the "index," "Address," and "Zip" columns are dropped as they are considered unnecessary for the analysis.

---

4. Checking Data Information

Python Code

```
df.info()
```

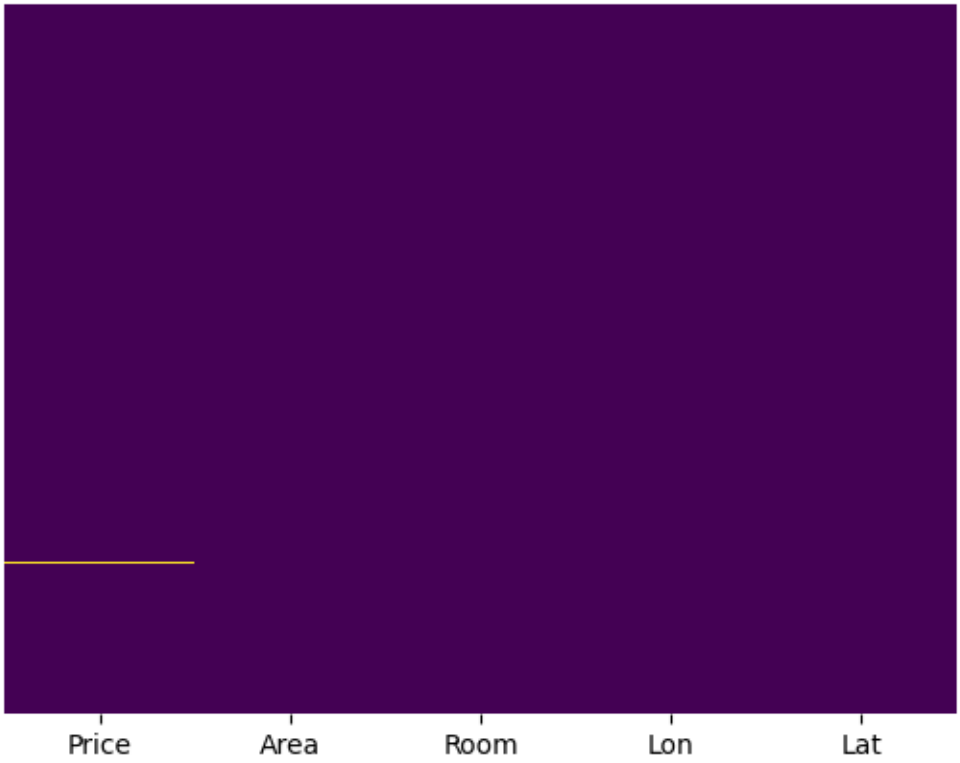
Data types and null values in the DataFrame are checked using the `.info()` method and displayed.

---

5. Visualizing Missing Values

Python Code

```
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```



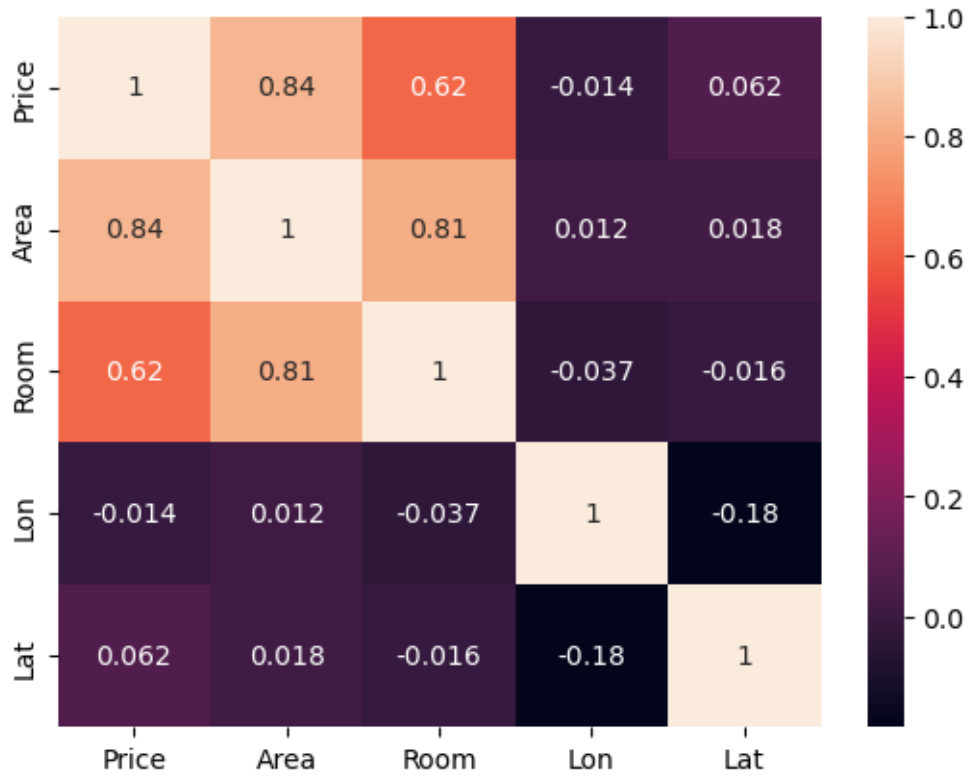
A heatmap is created to visualize missing values in the dataset using Seaborn's heatmap function.

---

6. Exploratory Data Analysis (EDA)

Python Code

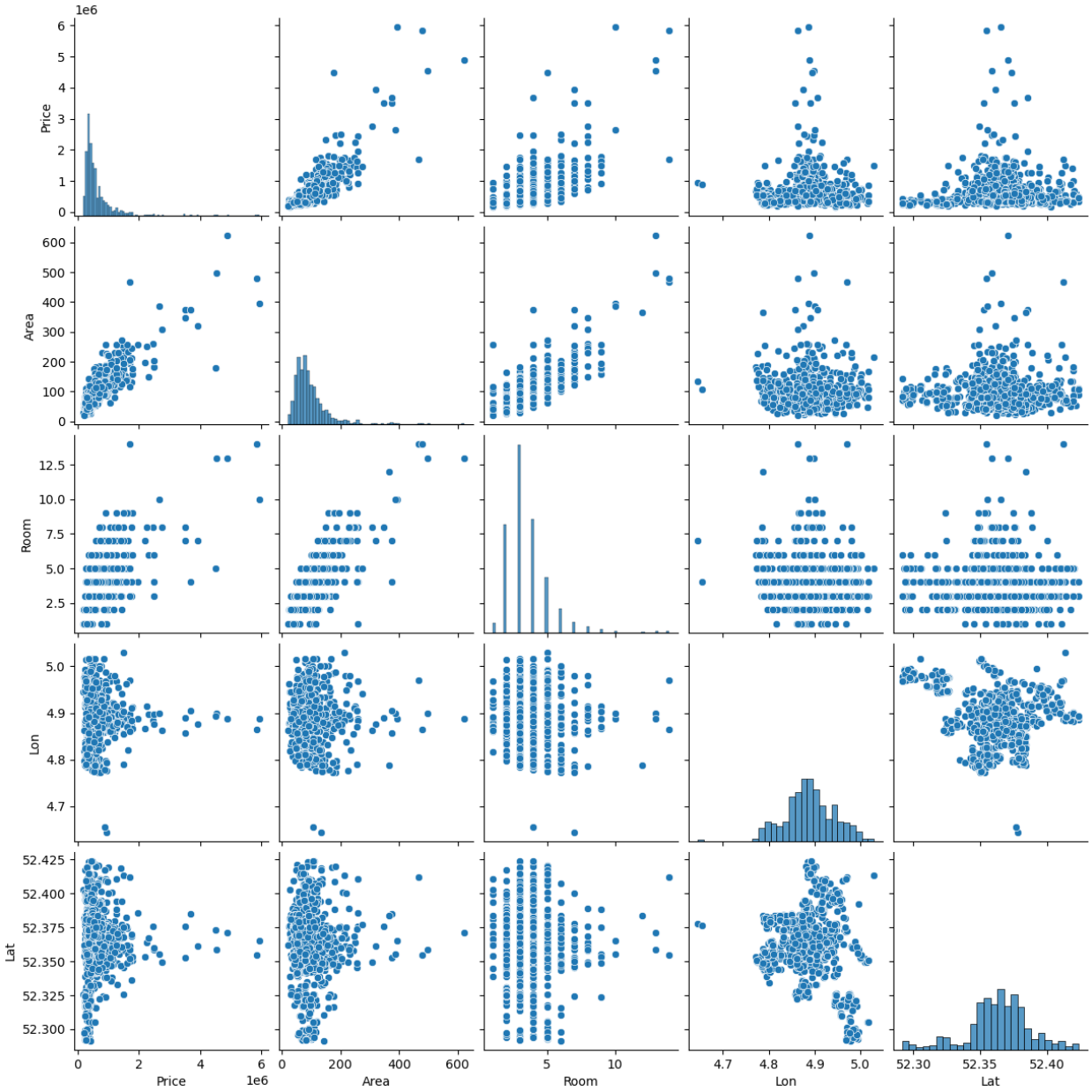
sns.heatmap(df.corr(), annot=True)



A heatmap is created to visualize the correlation between different columns in the dataset.

Python Code

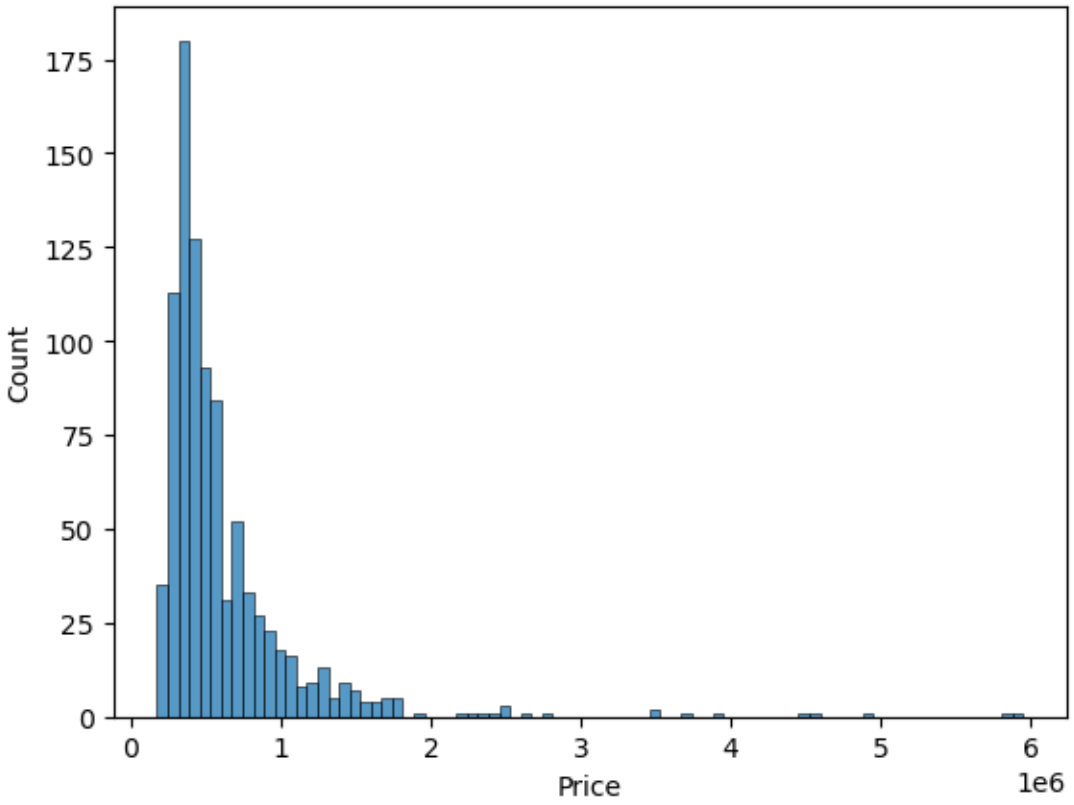
sns.pairplot(df)



A pairplot is generated to explore relationships between variables in the dataset.

Python Code

```
sns.histplot(df["Price"])
```



A histogram is created to visualize the distribution of prices in the dataset.

---

7. Data Normalization

Data normalization is performed on the "Price" column by removing outliers using the IQR method. Outliers are identified, the IQR is calculated, and a threshold for outliers is defined. Data points with prices above this threshold are removed.

Python code

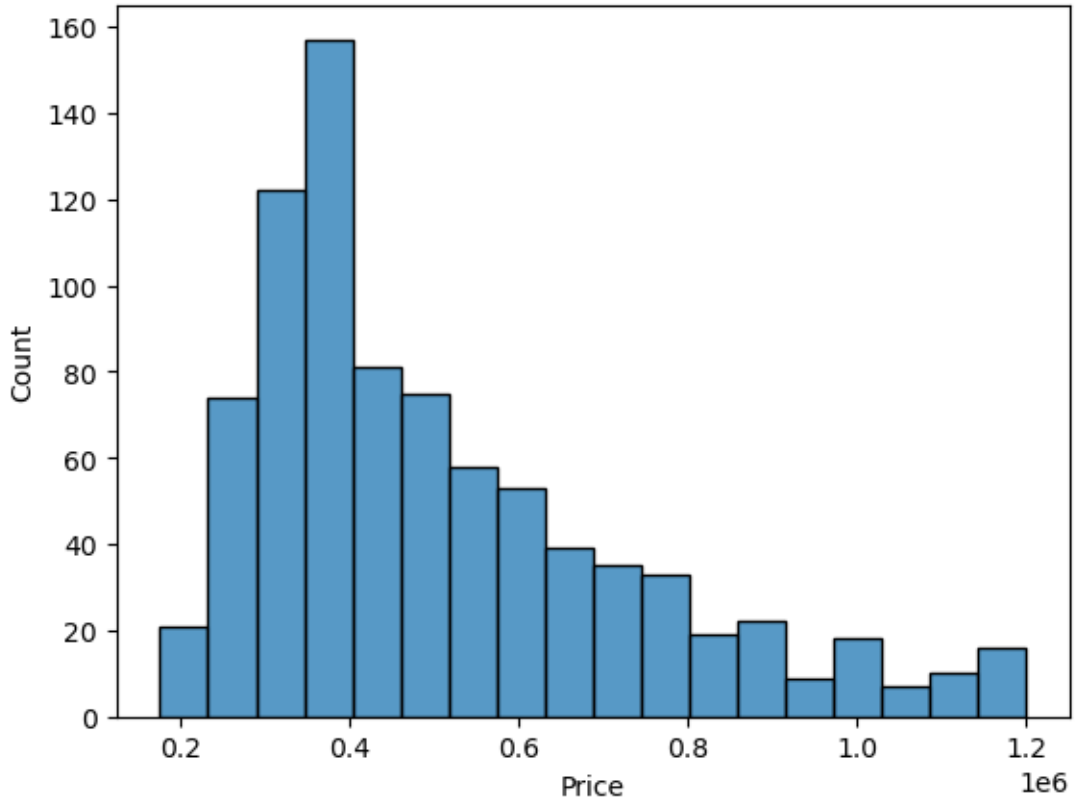
```
q1 = df.describe()['Price']['25%']

q3 = df.describe()['Price']['75%']

iqr = q3 - q1

max_price = q3 + 1.5 * iqr

max_price

outlier = df[df['Price'] >= max_price]

outliers_count = outlier['Price'].count()

data_count = df['Price'].count()

print('Percentage removed: ' + str(round(outliers_count/data_count * 100, 2)) + '%')

df= df[df['Price'] <= max_price]
```

---

8. Visualizing Normalized Data

Python Code

```
sns.histplot(df["Price"])
```



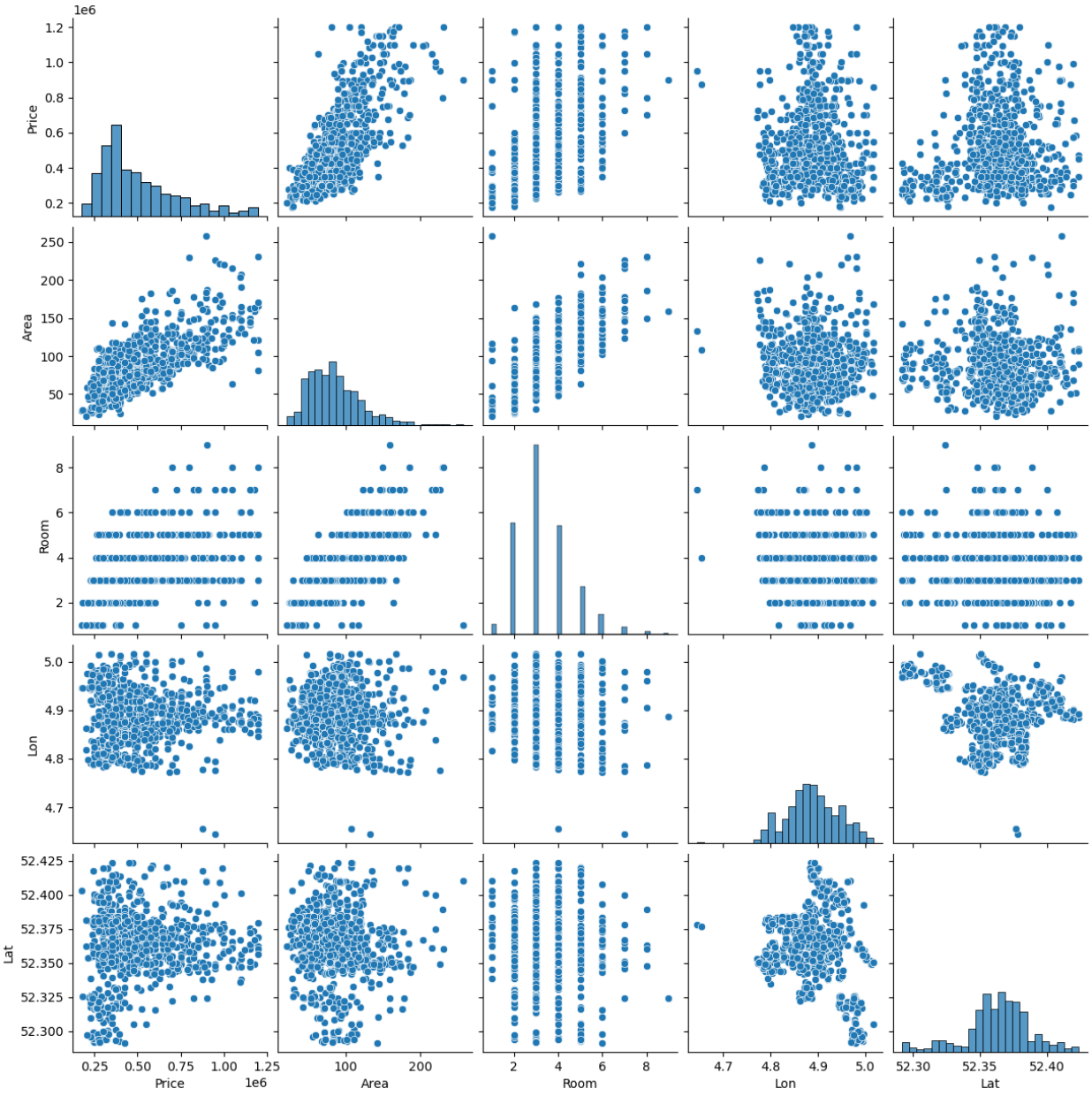
A histogram is created to visualize the distribution of prices after normalizing the "Price" column.

---

9. Further EDA

Python Code

sns.pairplot(df)



Another pairplot is generated to explore relationships between variables after removing outliers.

---

10. Splitting the Data

The data is split into features (X) and the target variable (Y) and prepared for model training.

Python Code

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

---

11. Model Training

Python Code

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()

lm.fit(X_train, y_train)
```

A linear regression model is trained using scikit-learn.

---

12. Model Evaluation

Python Code

```
print(lm.intercept_)
```

The intercept of the linear regression model is printed.

13. Interpreting the Coefficients

Python Code

```
coef=pd.DataFrame(lm.coef_,X.columns,columns=["Coefficient"])
coef
```

	Coefficient
Area	5.052650e+03
Room	-1.643080e+04
Lon	-1.191631e+05
Lat	1.204363e+06

1. Area Coefficient (5.052650e+03): This coefficient suggests that, on average, for every unit increase in the area of the property the price is expected to increase by approximately 5052.65 units, holding other variables constant.
2. Room Coefficient (-1.643080e+04): This coefficient suggests that, on average, for every additional room in the property, the price is expected to decrease by approximately 16430.8 units, holding other variables constant. A negative coefficient here indicates that more rooms are associated with lower prices.
3. Lon Coefficient (-1.191631e+05): This coefficient suggests that, on average, for every unit increase in longitude, the price is expected to decrease by approximately 119163.1 units, assuming other variables are held constant. This implies that properties located further east (higher longitude values) tend to have lower prices.
4. Lat Coefficient (1.204363e+06): This coefficient suggests that, on average, for every unit increase in latitude, the price is expected to increase by approximately 1204363 units, assuming other variables are held constant. This implies that properties located further north (higher latitude values) tend to have higher prices

Explanations for the coefficients of the linear regression model are provided based on their values and meanings in the context of the analysis.

---

14. Model Prediction and Evaluation

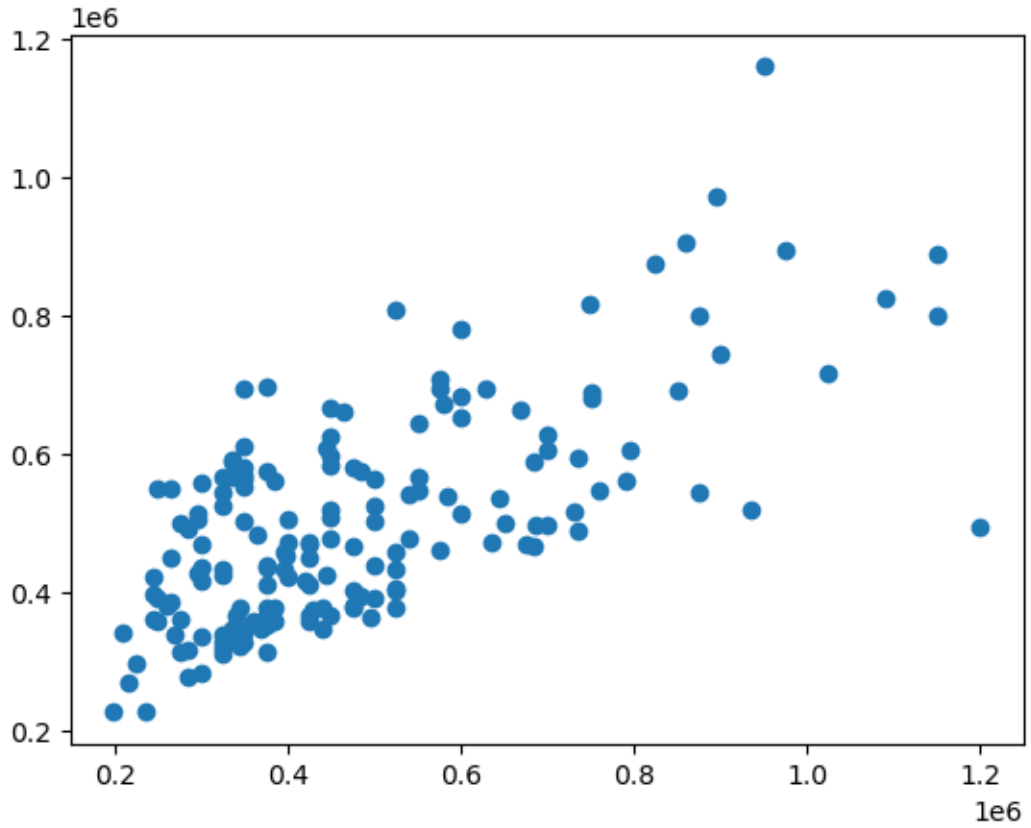
Python Code

```
prediction = lm.predict(X_test)
```

Predictions are made using the trained model on the test data.

Python Code

```
plt.scatter(y_test, prediction)
```



A scatter plot is created to visualize the relationship between the actual and predicted prices.

15. Regression Evaluation Metrics

```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, prediction))

print('MSE:', metrics.mean_squared_error(y_test, prediction))

print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```

```
MAE: 117560.57195933264
MSE: 23824560452.8607
RMSE: 154352.06656491742
```

Various regression metrics are calculated and printed to evaluate the performance of the model.

This documentation provides a comprehensive overview of the Jupyter Notebook, including data preprocessing, exploratory data analysis, model training, and evaluation. It also includes interpretations of coefficients and visualizations for better understanding.