**1. Handling Missing Values :**

**Bathrooms and Bedrooms :** The missing values in the 'bathrooms' and 'bedrooms' columns were filled based on the category of the listing. If it's a rental property, 'bathrooms' were filled with 1 and 'bedrooms' with 1; if it's an apartment, 'bathrooms' were filled with 2 and 'bedrooms' with 2; for other housing types, 'bathrooms' were filled with 3 and 'bedrooms' with 3.

**Amenities :** Mode values for 'amenities' were calculated separately for each category ('rent', 'apartment', 'housing'). Missing values in 'amenities' were then filled based on the mode of the respective category.

**Pets Allowed :** Missing values were filled with 'None'.

**City Name** **:** Missing values were filled with the mode (most frequent value) of the column.

**State** **:** Missing values were filled with the mode of the column.

**Address** **:** Addresses were filled based on the mode addresses for each city. If there's only one address for a city, that address is used; otherwise, the mode address for the city is used. If mode address isn't available, a default value ('unknown') is used.

**Longitude and Latitude** **:** Missing values in 'longitude' and 'latitude' columns were filled with the mean of the respective column. Additionally, negative values in the 'longitude' column were made positive using the absolute function (**abs()**), ensuring all longitude values are positive.

**2. Handling Categorical Data :**

**Label Encoding :** Categorical columns were identified, and label encoding was applied using `LabelEncoder()` from `sklearn.preprocessing`. This transforms categorical values into numerical labels, which is necessary for machine learning algorithms to operate on them.

These preprocessing techniques ensure that the dataset is clean, filled with meaningful values, and properly formatted for training machine learning models. Additionally, the techniques applied maintain the integrity and relevance of the data for accurate modeling and predictions.

**Data Splitting:**

By using “train\_test\_split” function from scikit-learn library to split the data into training and testing sets with test size = 30%.

**Model Selection:**

1. **Linear Regression:** Trained the linear regression model on the training data X\_train, Y\_train using the **fit()** method and generated predictions on the test data X\_test using the trained model's **predict()** method. The mean square error obtained from the model evaluation after setting the **random state** by **50** was **551916.432467397**.
2. **Polynomial Regression**

This code demonstrates polynomial regression, a type of regression analysis in which the relationship between the independent variable X and the dependent variable Y is modeled as an

nth degree polynomial.

**Sizes of your training & testing sets:**

Test size: 30%.

Train size:70%

**Techniques that were used to improve the results:**

* Set different degrees for the model to get the best accuracy and the lowest error, the best degree was 2.
* Set different values to the parameter random state the best one was 50.

**Comparison:**

|  |  |  |
| --- | --- | --- |
|  | Linear Regression | Polynomial Regression |
| MSE |  |  |
| R2 Score |  |  |

**Conclusion**:

The Polynomial Regression model performs better than the Linear Regression model. The Polynomial Regression model has a lower Mean Square Error and a higher R2 score.