Let's go through the steps to build a linear regression model using the dataset provided. Below is a detailed step-by-step breakdown of the code.

### Step 1: Import the Necessary Libraries

```python

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import OneHotEncoder, StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import joblib

```

### Step 2: Load the Dataset

```python

file\_path = '/mnt/data/Chennai.csv'

df = pd.read\_csv(file\_path)

```

### Step 3: Perform Exploratory Data Analysis (EDA)

1. \*\*First few rows of the dataset:\*\*

```python

print("First few rows of the dataset:")

print(df.head())

```

2. \*\*Summary statistics:\*\*

```python

print("\nSummary statistics:")

print(df.describe())

```

3. \*\*Information about the dataset:\*\*

```python

print("\nInformation about the dataset:")

print(df.info())

```

4. \*\*Check for missing values:\*\*

```python

print("\nMissing values in each column:")

print(df.isnull().sum())

```

5. \*\*Correlation matrix for numeric columns:\*\*

```python

numeric\_df = df.select\_dtypes(include=['float64', 'int64'])

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

sns.heatmap(numeric\_df.corr(), annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

```

6. \*\*Pairplot for numeric columns:\*\*

```python

sns.pairplot(numeric\_df)

plt.show()

```

### Step 4: Prepare the Data for Modeling

1. \*\*Select relevant columns:\*\*

```python

data\_selected = df[['Price', 'Area', 'No. of Bedrooms', 'Location']].copy()

```

2. \*\*Rename 'No. of Bedrooms' to 'Bedrooms':\*\*

```python

data\_selected.rename(columns={'Price': 'RentPrice', 'No. of Bedrooms': 'Bedrooms'}, inplace=True)

```

3. \*\*Handle missing values:\*\*

```python

data\_selected.dropna(inplace=True)

```

4. \*\*Define the features (X) and target (y):\*\*

```python

X = data\_selected.drop('RentPrice', axis=1)

y = data\_selected['RentPrice']

```

### Step 5: Split the Dataset into Training and Testing Sets

```python

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

```

### Step 6: Define the Preprocessing Steps

1. \*\*Categorical and numerical features:\*\*

```python

categorical\_features = ['Location']

numerical\_features = ['Area', 'Bedrooms']

```

2. \*\*Preprocessor for handling different types of data:\*\*

```python

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), numerical\_features),

('cat', OneHotEncoder(handle\_unknown='ignore'), categorical\_features)

]

)

```

### Step 7: Create the Model Pipeline

```python

model = Pipeline(steps=[

('preprocessor', preprocessor),

('regressor', LinearRegression())

])

```

### Step 8: Train the Model

```python

model.fit(X\_train, y\_train)

```

### Step 9: Save the Pipeline to a File

```python

joblib.dump(model, 'rent\_price\_model.pkl')

```

### Step 10: Predict on the Test Set

```python

y\_pred = model.predict(X\_test)

```

### Step 11: Evaluate the Model

1. \*\*Calculate RMSE, MAE, and R²:\*\*

```python

rmse = mean\_squared\_error(y\_test, y\_pred, squared=False)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f'RMSE: {rmse}')

print(f'MAE: {mae}')

print(f'R^2: {r2}')

```

### Step 12: Save the Trained Model

```python

joblib.dump(model, '11house\_rent\_prediction\_model.pkl')

```

This sequence of steps provides a comprehensive approach to building and evaluating a linear regression model for predicting rent prices. You can run this code with the provided dataset to train the model and assess its performance.