Assessment-3

Name:Damini N Reg number:20MID0119

1. Download the dataset: Dataset
2. Load the dataset into the tool.

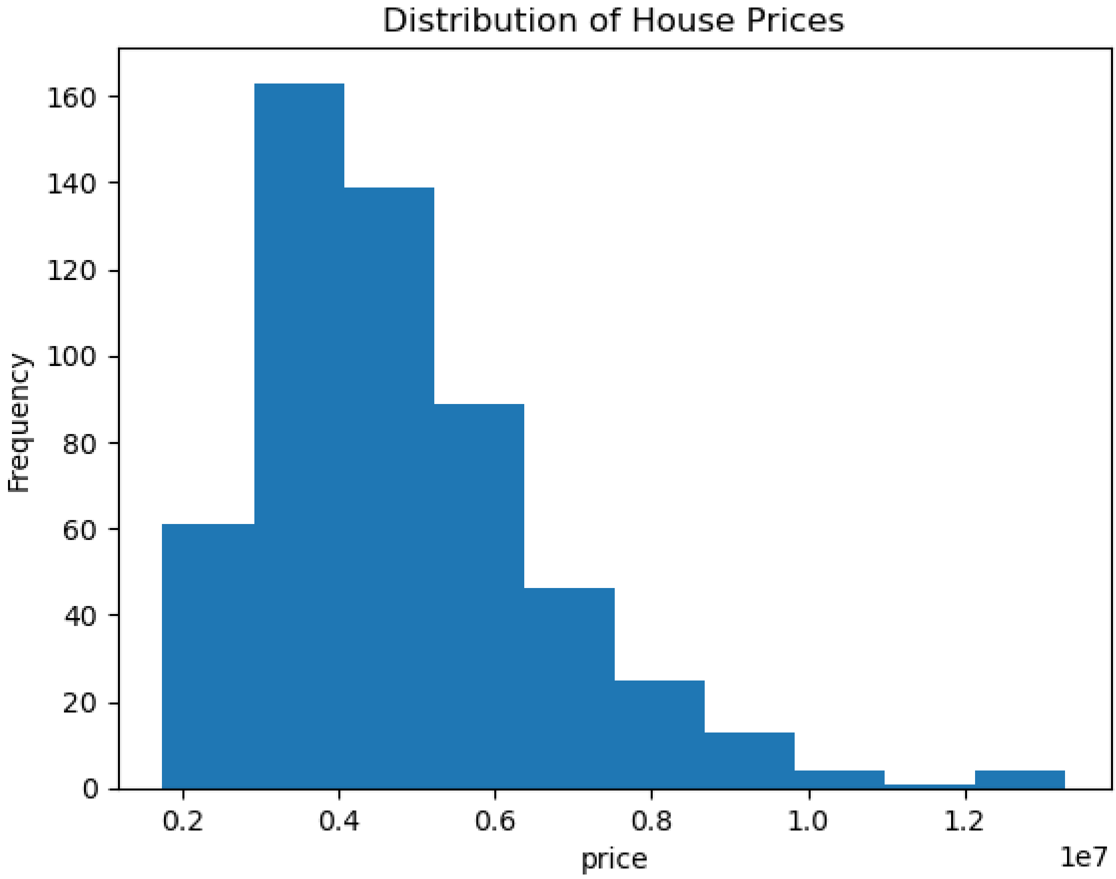
In [3]:

```
import pandas as pd

# Load the dataset into a pandas DataFrame
data = pd.read_csv("C:/Users/Damini N/Downloads/Housing.csv")
```

3. Perform Below Visualizations. □ Univariate Analysis □ Bi-Variate Analysis □ Multi-Variate Analysis

In [5]:

```
#Univariate analysis
import matplotlib.pyplot as plt

# Histogram of the 'Price' variable
plt.hist(data['price'], bins=10)
plt.xlabel('price')
plt.ylabel('Frequency')
plt.title('Distribution of House Prices')
plt.show()
```

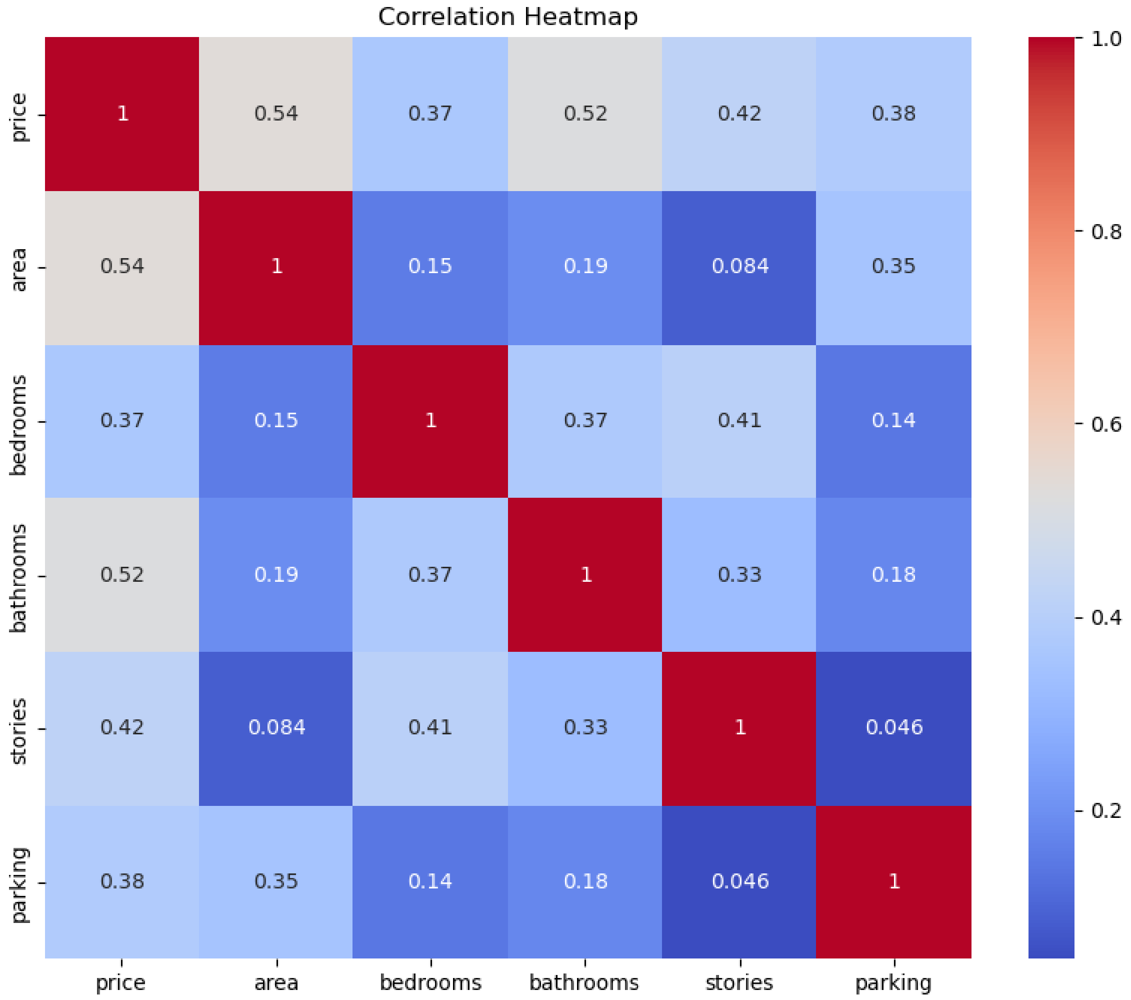


In [8]:

```
#Bivariate Analysis
# Scatter plot of 'Price' vs 'Area'
plt.scatter(data['area'], data['price'])
plt.xlabel('area')
plt.ylabel('price')
plt.title('Price vs Area')
plt.show()
```

In [9]:

```python
#Multivariate Analysis
import seaborn as sns

# Compute the correlation matrix
corr_matrix = data.corr()

# Generate a heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



4. Perform descriptive statistics on the dataset.

In [11]:

```
import pandas as pd

# Load the dataset into a pandas DataFrame
data = pd.read_csv("C:/Users/Damini N/Downloads/Housing.csv")

# Calculate descriptive statistics
descriptive_stats = data.describe()

# Print the descriptive statistics
print(descriptive_stats)
```

```
              price          area    bedrooms   bathrooms     stories  \
count  5.450000e+02    545.000000  545.000000  545.000000  545.000000
mean   4.766729e+06   5150.541284    2.965138    1.286239    1.805505
std    1.870440e+06   2170.141023    0.738064    0.502470    0.867492
min    1.750000e+06   1650.000000    1.000000    1.000000    1.000000
25%    3.430000e+06   3600.000000    2.000000    1.000000    1.000000
50%    4.340000e+06   4600.000000    3.000000    1.000000    2.000000
75%    5.740000e+06   6360.000000    3.000000    2.000000    2.000000
max    1.330000e+07  16200.000000    6.000000    4.000000    4.000000

          parking
count  545.000000
mean     0.693578
std      0.861586
min      0.000000
25%      0.000000
50%      0.000000
75%      1.000000
max      3.000000
```

5.Check for Missing values and deal with them.

In [13]:

```
import pandas as pd

# Load the dataset into a pandas DataFrame
data = pd.read_csv("C:/Users/Damini N/Downloads/Housing.csv")

# Check for missing values
missing_values = data.isnull().sum()
print(missing_values)
```

```
price               0
area                0
bedrooms            0
bathrooms           0
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
furnishingstatus    0
dtype: int64
```

In [14]:

```
# Remove rows with missing values
data = data.dropna()
```

In [16]:

```
# Impute missing values with the mean
data['area'].fillna(data['area'].mean(), inplace=True)

# Impute missing values with the median
data['bedrooms'].fillna(data['bedrooms'].median(), inplace=True)
```

In [18]:

```
# Encode missing values as a separate category
data['furnishingstatus'].fillna('unknown', inplace=True)

# Assign missing values the most frequent category
most_frequent_category = data['basement'].mode()[0]
data['basement'].fillna(most_frequent_category, inplace=True)
```

6. Find the outliers and replace them outliers

In [20]:

```
import pandas as pd
import numpy as np

# Load the dataset into a pandas DataFrame
data = pd.read_csv("C:/Users/Damini N/Downloads/Housing.csv")

# Calculate z-scores for each numerical variable
z_scores = (data[['price', 'area', 'bedrooms', 'bathrooms', 'stories']] - data[['price',

# Set the threshold for identifying outliers (e.g., z-score greater than 3)
threshold = 3

# Identify outliers based on z-scores
outliers = np.abs(z_scores) > threshold

# Replace outliers with NaN or other appropriate values
data[outliers] = np.nan
```

7. Check for Categorical columns and perform encoding.

In [21]:

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Load the dataset into a pandas DataFrame
data = pd.read_csv("C:/Users/Damini N/Downloads/Housing.csv")

# Identify categorical columns
categorical_columns = data.select_dtypes(include=['object']).columns

# Perform one-hot encoding on categorical columns
encoder = OneHotEncoder(sparse=False, drop='first')  # Initialize the encoder
encoded_data = pd.DataFrame(encoder.fit_transform(data[categorical_columns]))  # Perform
encoded_data.columns = encoder.get_feature_names(categorical_columns)  # Set column name

# Concatenate the encoded data with the original data
data_encoded = pd.concat([data.drop(categorical_columns, axis=1), encoded_data], axis=1)
```

```
C:\Users\Damini N\anaconda3\lib\site-packages\sklearn\utils\deprecation.p
y:87: FutureWarning: Function get_feature_names is deprecated; get_feature
_names is deprecated in 1.0 and will be removed in 1.2. Please use get_fea
ture_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

8. Split the data into dependent and independent variables.

In [22]:

```
import pandas as pd

# Load the dataset into a pandas DataFrame
data = pd.read_csv("C:/Users/Damini N/Downloads/Housing.csv")

# Split the data into dependent (target) variable and independent variables
X = data.drop('price', axis=1)  # Independent variables (all columns except 'Price')
y = data['price']  # Dependent variable ('Price')
```

9. Scale the independent variables

In [36]:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load the dataset into a pandas DataFrame
data = pd.read_csv("C:/Users/Damini N/Downloads/Housing.csv")

# Split the data into dependent (target) variable and independent variables
X = data.drop(['price', 'furnishingstatus'], axis=1)  # Independent variables excluding
y = data['price']  # Dependent variable ('Price')

# Identify categorical columns
categorical_columns = ['furnishingstatus']

# Perform one-hot encoding on categorical columns
encoded_data = pd.get_dummies(data[categorical_columns], drop_first=True)

# Concatenate the encoded data with the original data
X_encoded = pd.concat([X, encoded_data], axis=1)

# Scale the numerical independent variables
numerical_columns = X_encoded.select_dtypes(include=['float64', 'int64']).columns
scaler = StandardScaler()
X_encoded[numerical_columns] = scaler.fit_transform(X_encoded[numerical_columns])
```

10. Split the data into training and testing

In [30]:

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the dataset into a pandas DataFrame
data = pd.read_csv("C:/Users/Damini N/Downloads/Housing.csv")

# Split the data into dependent (target) variable and independent variables
X = data.drop('price', axis=1)  # Independent variables (all columns except 'Price')
y = data['price']  # Dependent variable ('Price')

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

11. Build the Model

In [40]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder

# Load the dataset into a pandas DataFrame
data = pd.read_csv("C:/Users/Damini N/Downloads/Housing.csv")

# Split the data into dependent (target) variable and independent variables
X = data.drop('price', axis=1)  # Independent variables
y = data['price']  # Dependent variable

# Identify categorical columns
categorical_columns = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'aircondi

# Perform one-hot encoding on categorical columns
encoder = OneHotEncoder(drop='first')
X_encoded = pd.DataFrame(encoder.fit_transform(X[categorical_columns]).toarray(),
                         columns=encoder.get_feature_names(categorical_columns))

# Concatenate the encoded data with the remaining independent variables
X = pd.concat([X.drop(categorical_columns, axis=1), X_encoded], axis=1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# Initialize the Linear Regression model
model = LinearRegression()

# Fit the model on the training data
model.fit(X_train, y_train)

# Predict on the testing data
y_pred = model.predict(X_test)

# Calculate the Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)
```

```
Mean Squared Error (MSE): 1837637189871.7024

C:\Users\Damini N\anaconda3\lib\site-packages\sklearn\utils\deprecation.p
y:87: FutureWarning: Function get_feature_names is deprecated; get_feature
_names is deprecated in 1.0 and will be removed in 1.2. Please use get_fea
ture_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

12. Train the Model

In [42]:

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder

# Load the dataset into a pandas DataFrame
data = pd.read_csv("C:/Users/Damini N/Downloads/Housing.csv")

# Split the data into dependent (target) variable and independent variables
X = data.drop('price', axis=1)  # Independent variables
y = data['price']  # Dependent variable

# Identify categorical columns
categorical_columns = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'aircondi

# Perform one-hot encoding on categorical columns
encoder = OneHotEncoder(drop='first')
X_encoded = pd.DataFrame(encoder.fit_transform(X[categorical_columns]).toarray(),
                         columns=encoder.get_feature_names(categorical_columns))

# Drop the original categorical columns from X
X = X.drop(categorical_columns, axis=1)

# Concatenate the encoded data with the remaining independent variables
X = pd.concat([X, X_encoded], axis=1)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# Initialize the Linear Regression model
model = LinearRegression()

# Fit the model on the training data
model.fit(X_train, y_train)

# Predict on the testing data
y_pred = model.predict(X_test)

# Calculate the Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)
```

```
Mean Squared Error (MSE): 1837637189871.7024

C:\Users\Damini N\anaconda3\lib\site-packages\sklearn\utils\deprecation.p
y:87: FutureWarning: Function get_feature_names is deprecated; get_feature
_names is deprecated in 1.0 and will be removed in 1.2. Please use get_fea
ture_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

13. Test the Model

In [43]:

```
from sklearn.metrics import mean_squared_error, r2_score

# Predict on the testing data
y_pred = model.predict(X_test)

# Calculate the Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)

# Calculate the R-squared score
r2 = r2_score(y_test, y_pred)
print("R-squared Score:", r2)
```

```
Mean Squared Error (MSE): 1837637189871.7024
R-squared Score: 0.6364404686639471
```

14. Measure the performance using Metrics.

In [44]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)

# Calculate Root Mean Squared Error (RMSE)
rmse = mean_squared_error(y_test, y_pred, squared=False)
print("Root Mean Squared Error (RMSE):", rmse)

# Calculate R-squared score
r2 = r2_score(y_test, y_pred)
print("R-squared Score:", r2)
```

```
Mean Absolute Error (MAE): 988116.163240571
Root Mean Squared Error (RMSE): 1355594.7734746186
R-squared Score: 0.6364404686639471
```

In [ ]: