hw_3_question_2

March 3, 2025

[1]:
```
import pandas as pd
```

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[3]:
```
# Read in the data set
df = pd.read_csv('/Users/helenamabey/Stats_Spring_2025/Real_estate.csv')
```

[4]:
```
df.head()
```

[4]:
```
   No  Transaction date  House age  Distance to the nearest MRT station  \
0   1          2012.917       32.0                             84.87882
1   2          2012.917       19.5                            306.59470
2   3          2013.583       13.3                            561.98450
3   4          2013.500       13.3                            561.98450
4   5          2012.833        5.0                            390.56840

   Number of convenience stores  Latitude  Longitude  House price of unit area
0                            10  24.98298  121.54024                      37.9
1                             9  24.98034  121.53951                      42.2
2                             5  24.98746  121.54391                      47.3
3                             5  24.98746  121.54391                      54.8
4                             5  24.97937  121.54245                      43.1
```

[5]:
```
# Used ChatGPT to help update the given date format to a usable date format.␣
↪Defined a function to capture a standard date as described
# in the initial data definition table in homework.
from datetime import datetime, timedelta

def decimal_year_to_date(decimal_year):
    year = int(decimal_year)
    remainder = decimal_year - year
    start_of_year = datetime(year, 1, 1)
    days_in_year = (datetime(year + 1, 1, 1) - start_of_year).days
    actual_date = start_of_year + timedelta(days=remainder * days_in_year)
    return actual_date.strftime("%Y-%m-%d")
```

```python
df['Transaction date'] = [decimal_year_to_date(d) for d in df['Transaction␣
↪date']]

df.head()
```

[5]:
```
   No Transaction date  House age  Distance to the nearest MRT station  \
0   1       2012-12-01       32.0                             84.87882
1   2       2012-12-01       19.5                            306.59470
2   3       2013-08-01       13.3                            561.98450
3   4       2013-07-02       13.3                            561.98450
4   5       2012-10-31        5.0                            390.56840

   Number of convenience stores  Latitude  Longitude  House price of unit area
0                            10  24.98298  121.54024                      37.9
1                             9  24.98034  121.53951                      42.2
2                             5  24.98746  121.54391                      47.3
3                             5  24.98746  121.54391                      54.8
4                             5  24.97937  121.54245                      43.1
```

[6]:
```python
# Review the properties of each column. The date is not the correct data type.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   No                                   414 non-null    int64
 1   Transaction date                     414 non-null    object
 2   House age                            414 non-null    float64
 3   Distance to the nearest MRT station  414 non-null    float64
 4   Number of convenience stores         414 non-null    int64
 5   Latitude                             414 non-null    float64
 6   Longitude                            414 non-null    float64
 7   House price of unit area             414 non-null    float64
dtypes: float64(5), int64(2), object(1)
memory usage: 26.0+ KB
```

[7]:
```python
# Correct the Transaction date data type
df['Transaction date'] = pd.to_datetime(df['Transaction date'])
df.head()
```

[7]:
```
   No Transaction date  House age  Distance to the nearest MRT station  \
0   1       2012-12-01       32.0                             84.87882
1   2       2012-12-01       19.5                            306.59470
2   3       2013-08-01       13.3                            561.98450
3   4       2013-07-02       13.3                            561.98450
```

```
4   5       2012-10-31        5.0                              390.56840

   Number of convenience stores  Latitude  Longitude  House price of unit area
0                            10  24.98298  121.54024                      37.9
1                             9  24.98034  121.53951                      42.2
2                             5  24.98746  121.54391                      47.3
3                             5  24.98746  121.54391                      54.8
4                             5  24.97937  121.54245                      43.1
```

[8]:
```python
# Confirmed the data type has been updated
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   No                                  414 non-null    int64
 1   Transaction date                    414 non-null    datetime64[ns]
 2   House age                           414 non-null    float64
 3   Distance to the nearest MRT station 414 non-null    float64
 4   Number of convenience stores        414 non-null    int64
 5   Latitude                            414 non-null    float64
 6   Longitude                           414 non-null    float64
 7   House price of unit area            414 non-null    float64
dtypes: datetime64[ns](1), float64(5), int64(2)
memory usage: 26.0 KB
```

[9]:
```python
# Obtain the summary statistics on the full data set
df.describe()
```

[9]:
```
               No   House age  Distance to the nearest MRT station  \
count  414.000000  414.000000                           414.000000
mean   207.500000   17.712560                          1083.885689
std    119.655756   11.392485                          1262.109595
min      1.000000    0.000000                            23.382840
25%    104.250000    9.025000                           289.324800
50%    207.500000   16.100000                           492.231300
75%    310.750000   28.150000                          1454.279000
max    414.000000   43.800000                          6488.021000

       Number of convenience stores    Latitude   Longitude  \
count                    414.000000  414.000000  414.000000
mean                       4.094203   24.969030  121.533361
std                        2.945562    0.012410    0.015347
min                        0.000000   24.932070  121.473530
25%                        1.000000   24.963000  121.528085
```

```
50%                        4.000000   24.971100  121.538630
75%                        6.000000   24.977455  121.543305
max                       10.000000   25.014590  121.566270

       House price of unit area
count                414.000000
mean                  37.980193
std                   13.606488
min                    7.600000
25%                   27.700000
50%                   38.450000
75%                   46.600000
max                  117.500000
```

0.1 House Age vs House Price of Unit Area: Question 2 #1

[10]:
```python
# Obtain the summary statistics on the requested comparison features, House age␣
 ↪and House price of unit area
df[['House age','House price of unit area']].describe()
```

[10]:
```
        House age  House price of unit area
count  414.000000                414.000000
mean    17.712560                 37.980193
std     11.392485                 13.606488
min      0.000000                  7.600000
25%      9.025000                 27.700000
50%     16.100000                 38.450000
75%     28.150000                 46.600000
max     43.800000                117.500000
```

[14]:
```python
# Compute correlation between age and price: This shows that while there is a␣
 ↪negative correlation between the two
# features, it is very small. House prices do fall as the age of a house␣
 ↪increases but it is not a strong factor.
correlation = df[['House age', 'House price of unit area']].corr()
correlation
```

[14]:
```
                          House age  House price of unit area
House age                  1.000000                 -0.210567
House price of unit area  -0.210567                  1.000000
```

[47]:
```python
# House age skewness: This slightly right-skewed result is nearly 0 so the␣
 ↪distribution appears to be relatively normal
df["House age"].skew()
```

[47]: 0.38292623077299737

[49]:
```python
# House age Kurtosis: This shows that the distribution has a flatter peak than␣
↪a normal distribution and possibly
# fewer outliers
df["House age"].kurt()
```

[49]: -0.8771201112290763

[51]:
```python
# House price Skewness: This result is more moderately right-skewed than age,␣
↪showing there may be some higher
# outliers causing the skew.
df["House price of unit area"].skew()
```

[51]: 0.5998525842660576

[53]:
```python
# House price Kurtosis: Because this result is higher than normal, the␣
↪distribution is more sharply peaked with
# quite a few outliers.
df["House price of unit area"].kurt()
```

[53]: 2.1790970477396163

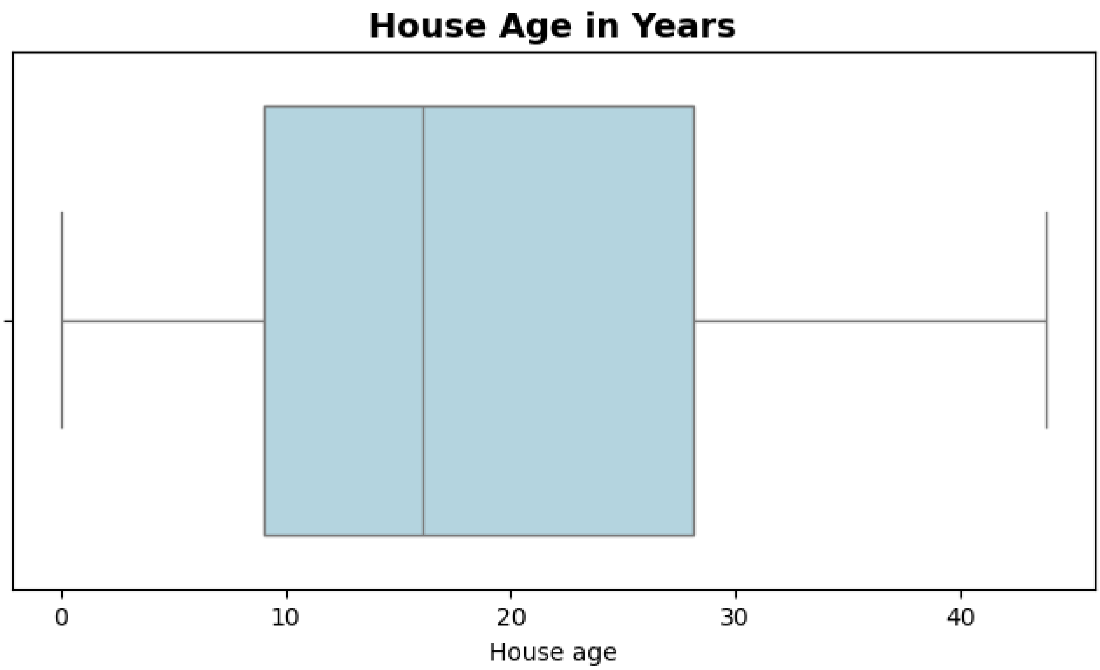
[55]:
```python
# Used ChatGPT for a code to calculate p value and Pearson's correlation (same␣
↪as correlation found above)
from scipy.stats import pearsonr

# Calculate Pearson correlation and p-value
corr_value, p_value = pearsonr(df['House age'], df['House price of unit area'])

print(f"Pearson's correlation: {corr_value:.3f}")
print(f"P-value: {p_value:.3f}")
```

```
Pearson's correlation: -0.211
P-value: 0.000
```

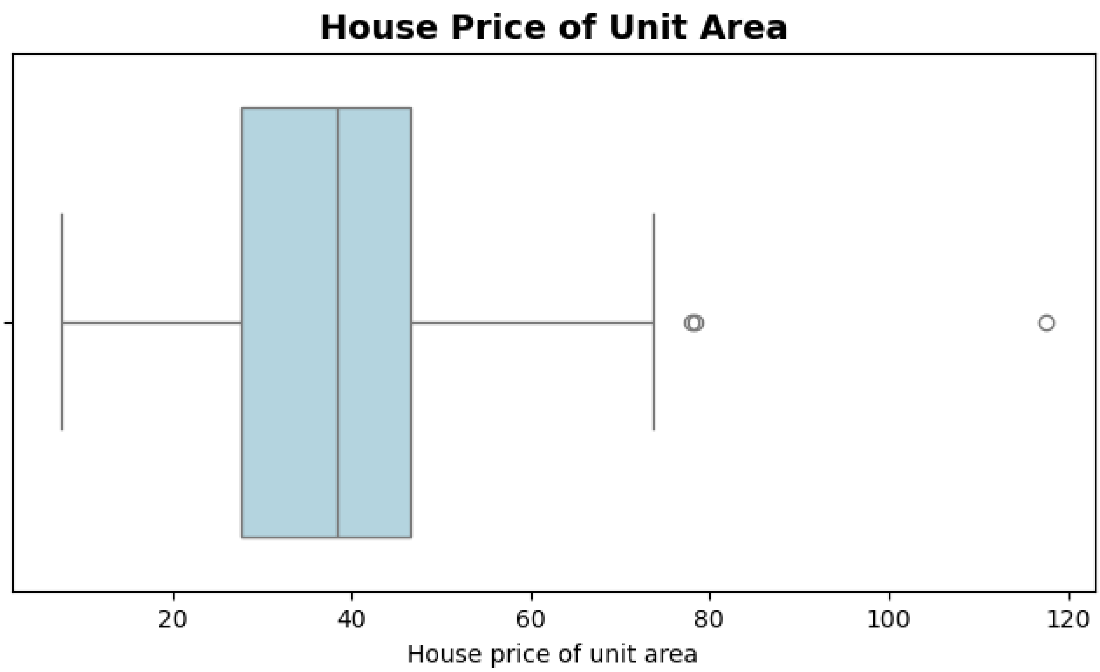
Both the mean and median are similar for House age and House price of unit area. This could show a normal distribution of the values. The correlation being a -0.210567 shows a very slight negative correlation between the age of houses in comparison to the price per unit. Older houses have a slightly lower price than newer homes. This confirmed with the Pearson's correlation. The P-value of 0.00 which is less than 0.05 also confirms that the house age does impact the price of unit area but again it shows the impact is very slight.

[159]:
```python
# Boxplot for Age: This confirms a slight right skew where the median value is␣
↪further toward the lower values
plt.figure(figsize=(8, 4))
sns.boxplot(x=df["House age"], color="lightblue")
plt.title("House Age in Years", fontsize=14, fontweight='bold')
```

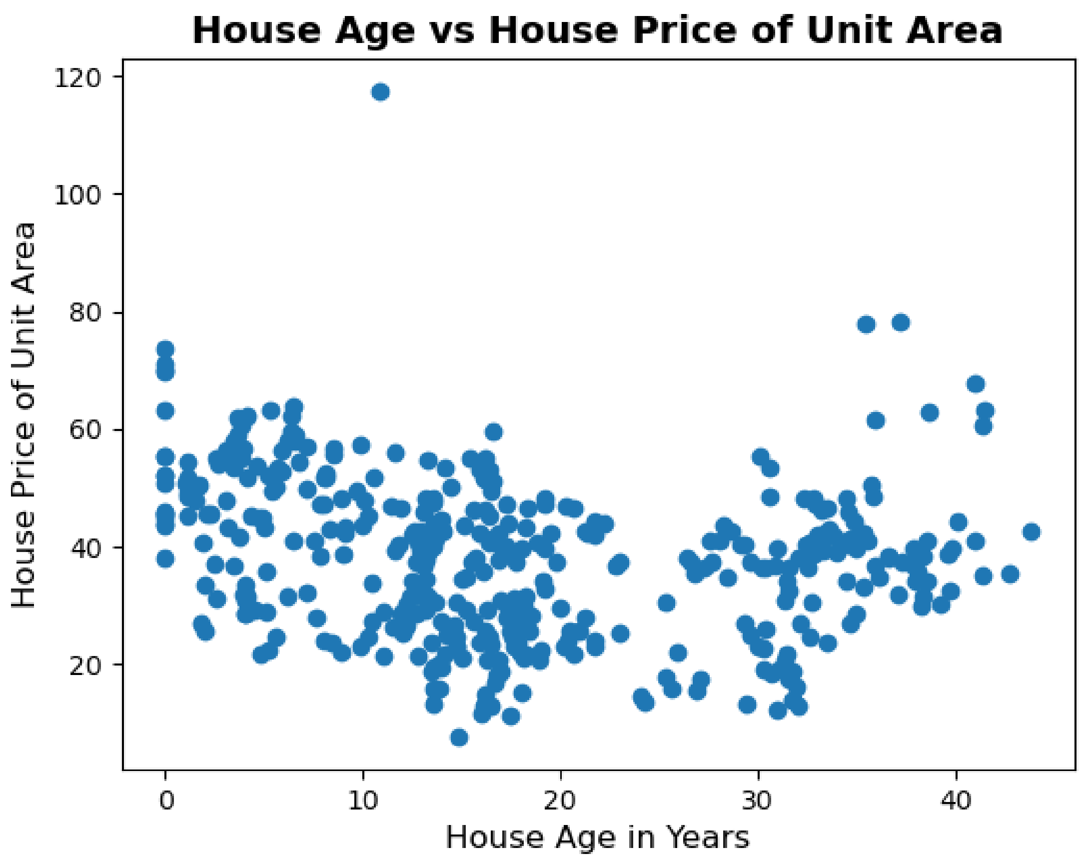
```
plt.show()
```



[157]:
```
# Boxplot for Price: This confirms that there is a more defined right-skew with␣
↪multiple large outliers on the far
# right. The skew is larger than for Age.
plt.figure(figsize=(8, 4))
sns.boxplot(x=df["House price of unit area"], color="lightblue")
plt.title("House Price of Unit Area", fontsize=14, fontweight='bold')
plt.show()
```



[155]:
```
#Scatterplot House age vs House price of unit area: This shows that there may␣
↪not be a strong relationship to age
# and price. There are a few outliers but generally the price results are␣
↪similar regardless of house age.
plt.scatter(x=df['House age'], y=df['House price of unit area'])
plt.title('House Age vs House Price of Unit Area', fontsize=14,␣
↪fontweight='bold')
plt.xlabel("House Age in Years", fontsize=12)
plt.ylabel("House Price of Unit Area", fontsize=12)
plt.show()
```



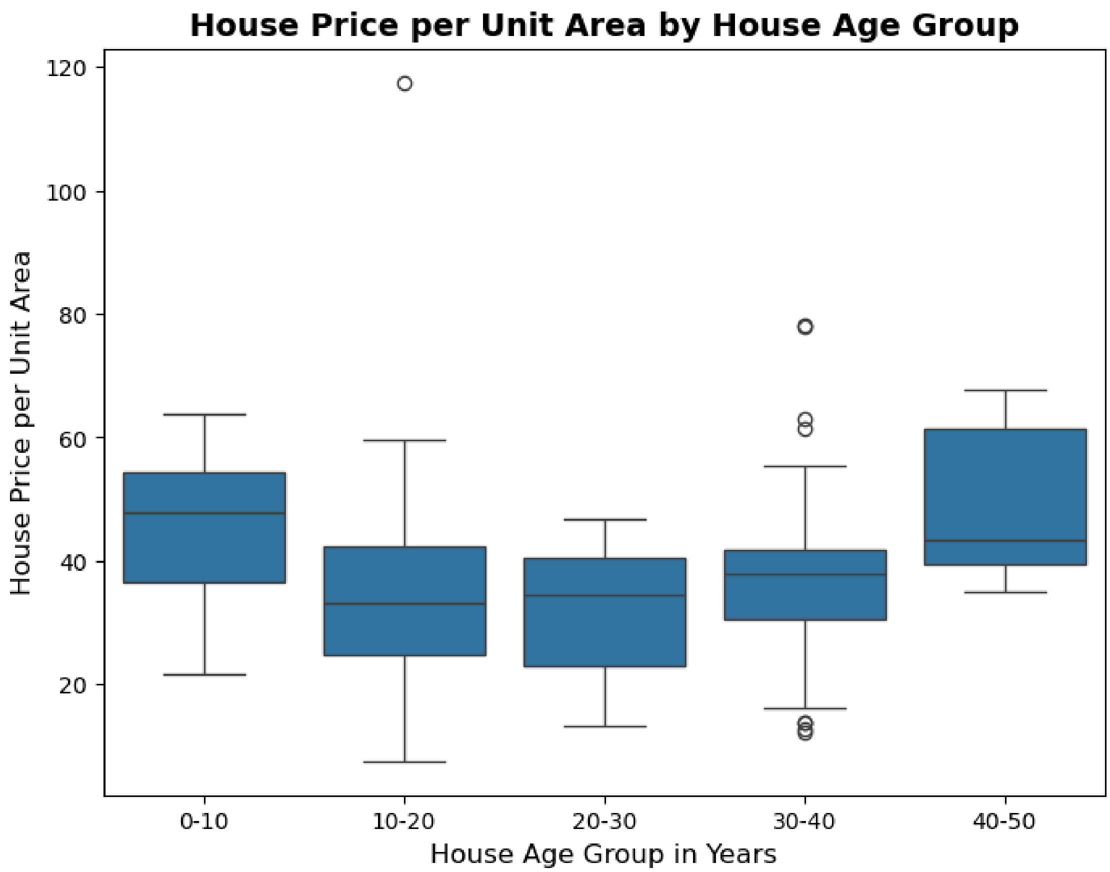
[17]:
```
pip install seaborn
```

```
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
                         294.9/294.9

kB 2.8 MB/s eta 0:00:00a 0:00:01
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from seaborn) (3.7.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from seaborn) (1.23.5)
Requirement already satisfied: pandas>=1.2 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from seaborn) (1.5.3)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
```

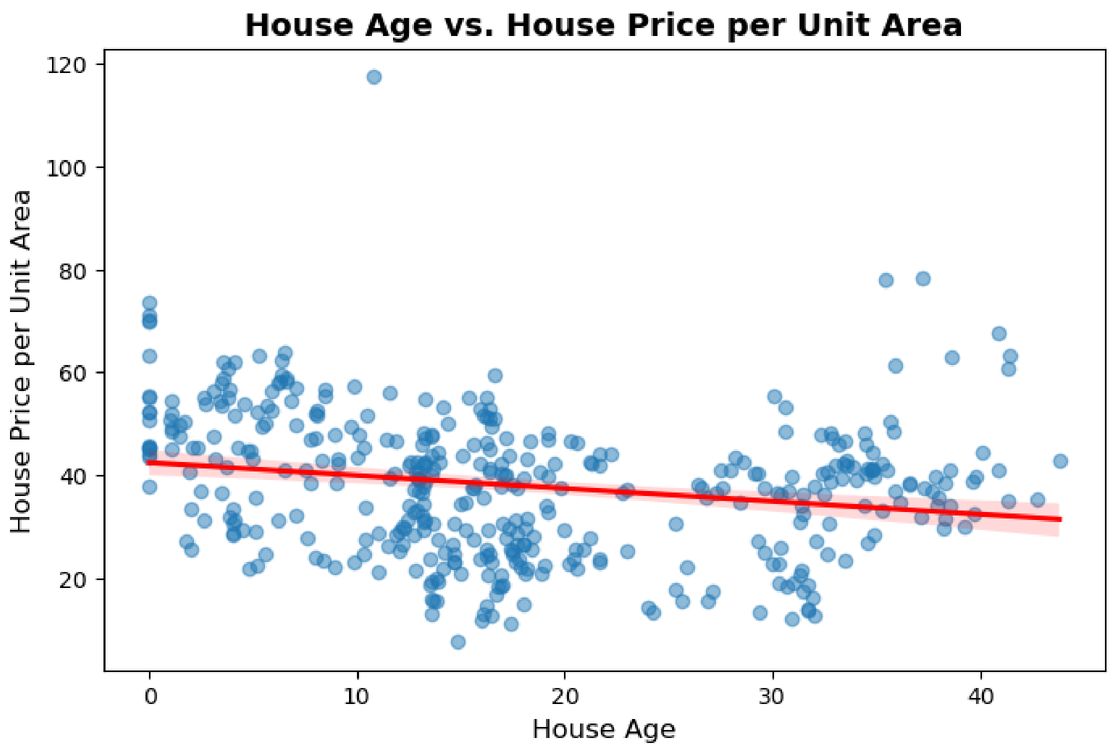
```
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.4)
Requirement already satisfied: cycler>=0.10 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (3.0.9)
Requirement already satisfied: pillow>=6.2.0 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (9.4.0)
Requirement already satisfied: packaging>=20.0 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (23.0)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (4.25.0)
Requirement already satisfied: contourpy>=1.0.1 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (1.0.5)
Requirement already satisfied: pytz>=2020.1 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from pandas>=1.2->seaborn) (2022.7)
Requirement already satisfied: six>=1.5 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2
Note: you may need to restart the kernel to use updated packages.
```

[19]:
```
import seaborn as sns
```

[153]:
```
# Using a multiple boxplot, this compares the price per unit area and the house␣
↪age: This also confirms the extreme outliers
# that may be impacting the normality of our data. It's interesting how once␣
↪the homes are considerably older, the prices increase.
df['House age category'] = pd.cut(df['House age'], bins=[0,10,20,30,40,50],␣
↪labels=['0-10','10-20','20-30','30-40','40-50'])

plt.figure(figsize=(8,6))
sns.boxplot(x='House age category', y='House price of unit area', data=df)
plt.xlabel("House Age Group in Years", fontsize=12)
plt.ylabel("House Price per Unit Area", fontsize=12)
plt.title("House Price per Unit Area by House Age Group", fontsize=14,␣
↪fontweight='bold')
plt.show()
```



[151]:
```
# Used ChatCPT and given python reference notebooks to assist with this␣
↪regression plot: This shows the slight negative relationship between
# these two factors. This confirms the correlation results we previously␣
↪reviewed that age has a minimal impact on price. The CI stays
# pretty thin meaning there is strong confidence in this trend even though␣
↪there is only a slight impact on price based on age.
plt.figure(figsize=(8, 5))
sns.regplot(x='House age', y='House price of unit area', data=df,␣
↪scatter_kws={'alpha':0.5}, line_kws={'color':'red'})
plt.xlabel("House Age", fontsize=12)
plt.ylabel("House Price per Unit Area", fontsize=12)
plt.title("House Age vs. House Price per Unit Area", fontsize=14,␣
↪fontweight='bold')
plt.show()
```



0.2 Distance to Nearest MRT Station vs House Price of Unit Area: Question 2 #2

[64]:
```
# Obtain the summary statistics on the requested comparison features, Distince␣
↪to nearest MRT station and price
df[['Distance to the nearest MRT station','House price of unit area']].
↪describe()
```

[64]:

	Distance to the nearest MRT station	House price of unit area
count	414.000000	414.000000
mean	1083.885689	37.980193
std	1262.109595	13.606488
min	23.382840	7.600000
25%	289.324800	27.700000
50%	492.231300	38.450000
75%	1454.279000	46.600000
max	6488.021000	117.500000

[99]:
```
# Compute correlation between distance and price: These results show a␣
↪relatively moderate negative correlation between
# these features. As the house price increases the distance decreases.
```

```
correlation_b = df[['Distance to the nearest MRT station', 'House price of unit area']].corr()
correlation_b
```

[99]:
```
                                     Distance to the nearest MRT station  \
Distance to the nearest MRT station                             1.000000
House price of unit area                                       -0.673613

                                     House price of unit area
Distance to the nearest MRT station                 -0.673613
House price of unit area                             1.000000
```

[68]:
```
# Distance Skewness: This results shows a pretty defined right-skew with many outliers on the right side.
df["Distance to the nearest MRT station"].skew()
```

[68]: 1.8887565801256048

[70]:
```
# Distance Kurtosis: This value is very high implying the distribution has a high peak with many outliers and heavy
# tails
df["Distance to the nearest MRT station"].kurt()
```

[70]: 3.20786836751181

[72]:
```
# House price Skewness: This result is more moderately right-skewed than age, showing there may be some higher
# outliers causing the skew.
df["House price of unit area"].skew()
```

[72]: 0.5998525842660576

[74]:
```
# House price Kurtosis: Because this result is higher than normal, the distribution is more sharply peaked with
# quite a few outliers.
df["House price of unit area"].kurt()
```

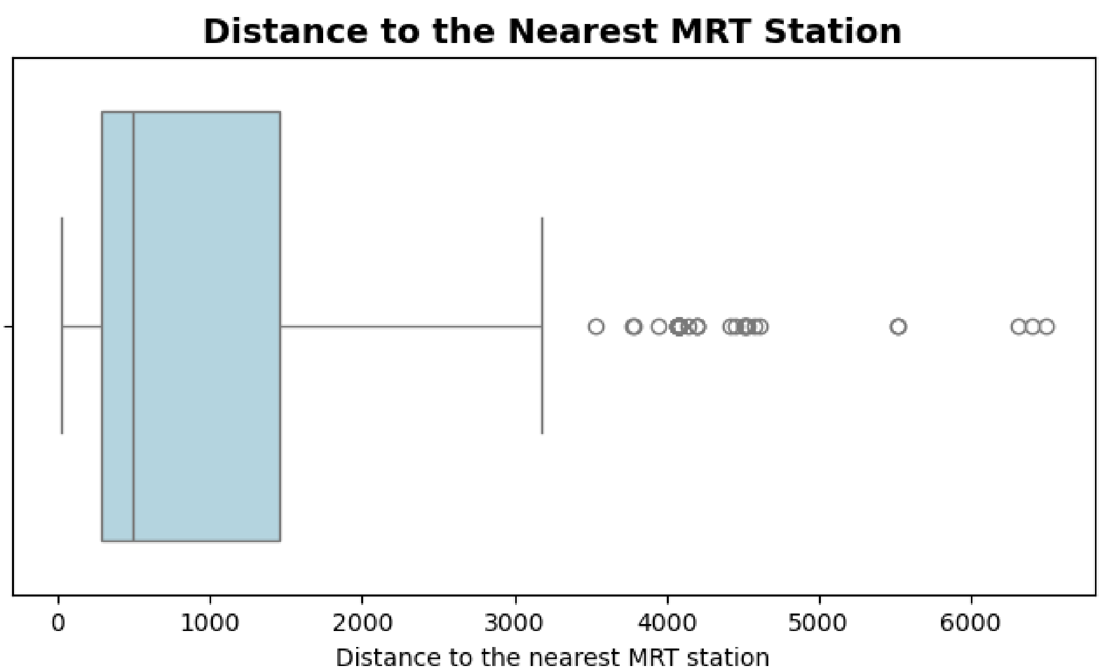
[74]: 2.1790970477396163

[111]:
```
# Used ChatGPT for a code to calculate p value and Pearson's correlation (same as correlation found above)
from scipy.stats import pearsonr

# Calculate Pearson correlation and p-value
corr_value_b, p_value_b = pearsonr(df['Distance to the nearest MRT station'], df['House price of unit area'])
```

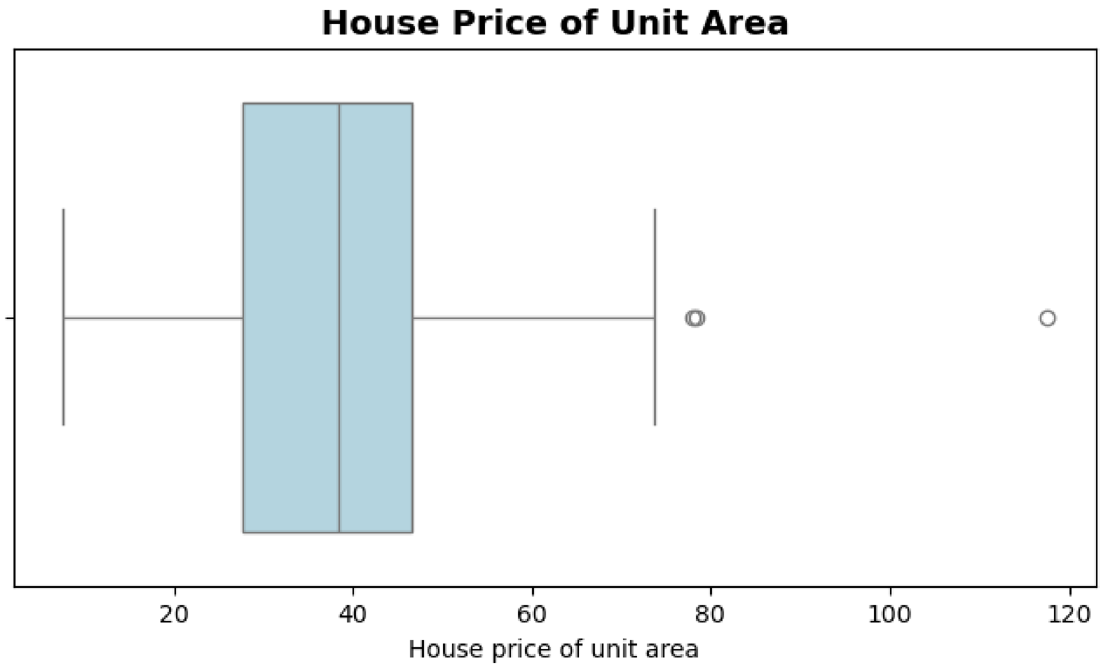
```
print(f"Pearson's correlation: {corr_value_b:.3f}")
print(f"P-value: {p_value_b:.3f}")
```

```
Pearson's correlation: -0.674
P-value: 0.000
```

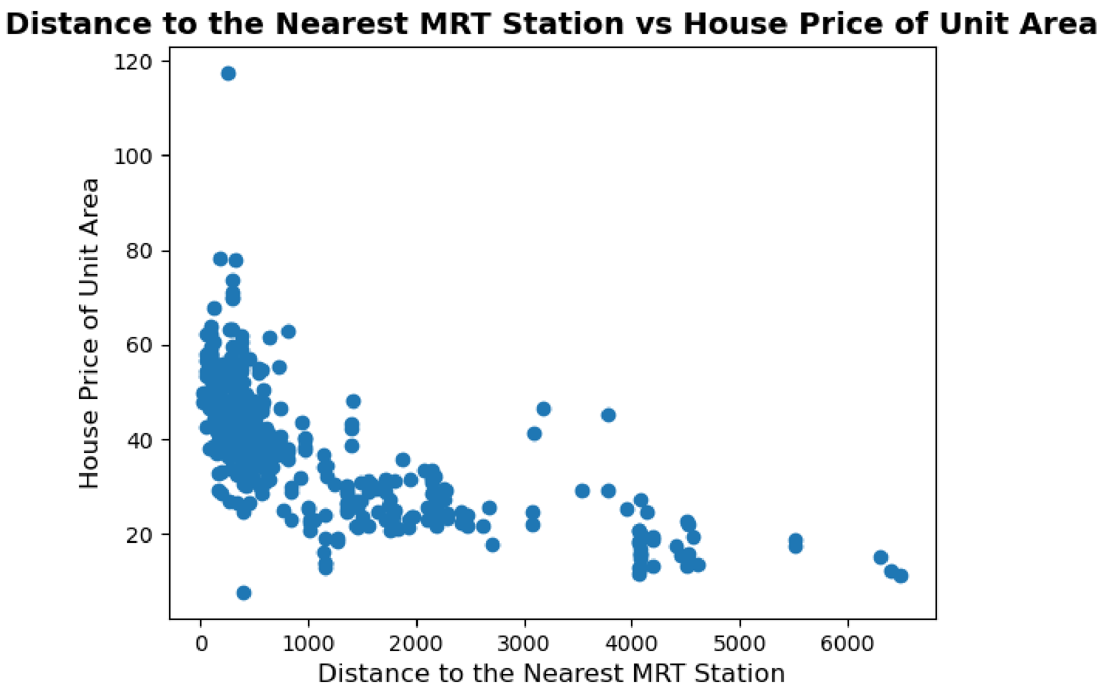
[149]:
```
# Boxplot for Distance: This clearly shows the right-skew of the data with many␣
 ↪outliers impacting the distribution.
plt.figure(figsize=(8, 4))
sns.boxplot(x=df["Distance to the nearest MRT station"], color="lightblue")
plt.title("Distance to the Nearest MRT Station", fontsize=14, fontweight='bold')
plt.show()
```



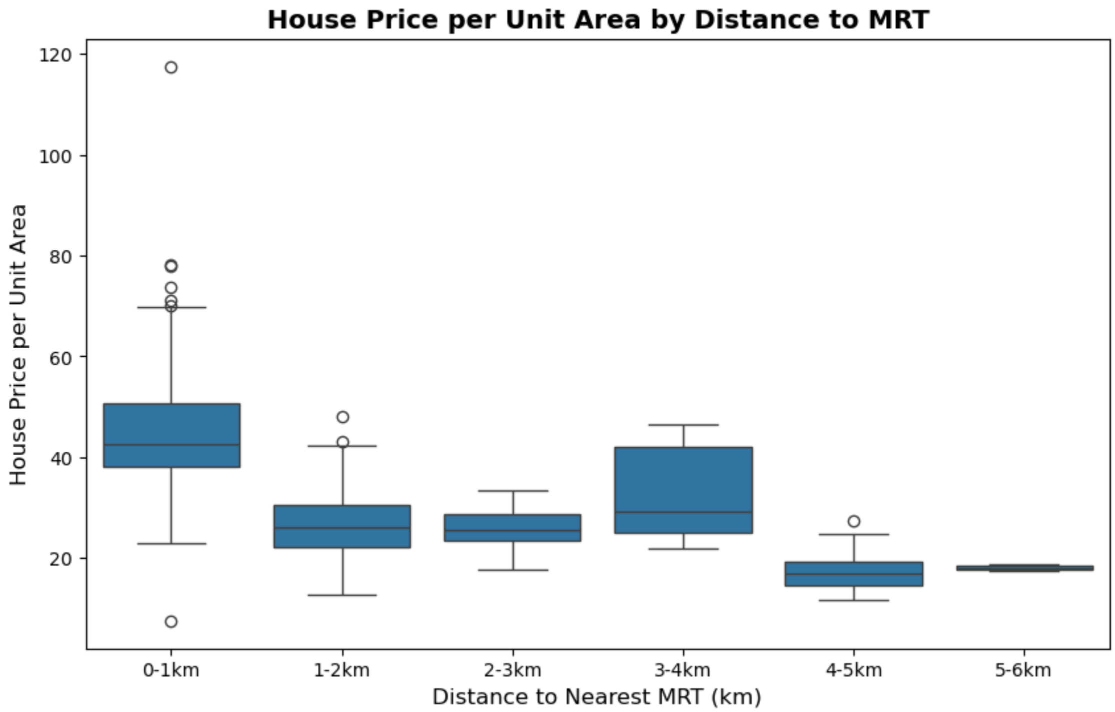
[147]:
```
# Boxplot for Price: This confirms that there is a more defined right-skew with␣
 ↪ multiple large outliers on the far
# right. This is much more moderately skewed than distance.
plt.figure(figsize=(8, 4))
sns.boxplot(x=df["House price of unit area"], color="lightblue")
plt.title("House Price of Unit Area", fontsize=14, fontweight='bold')
plt.show()
```



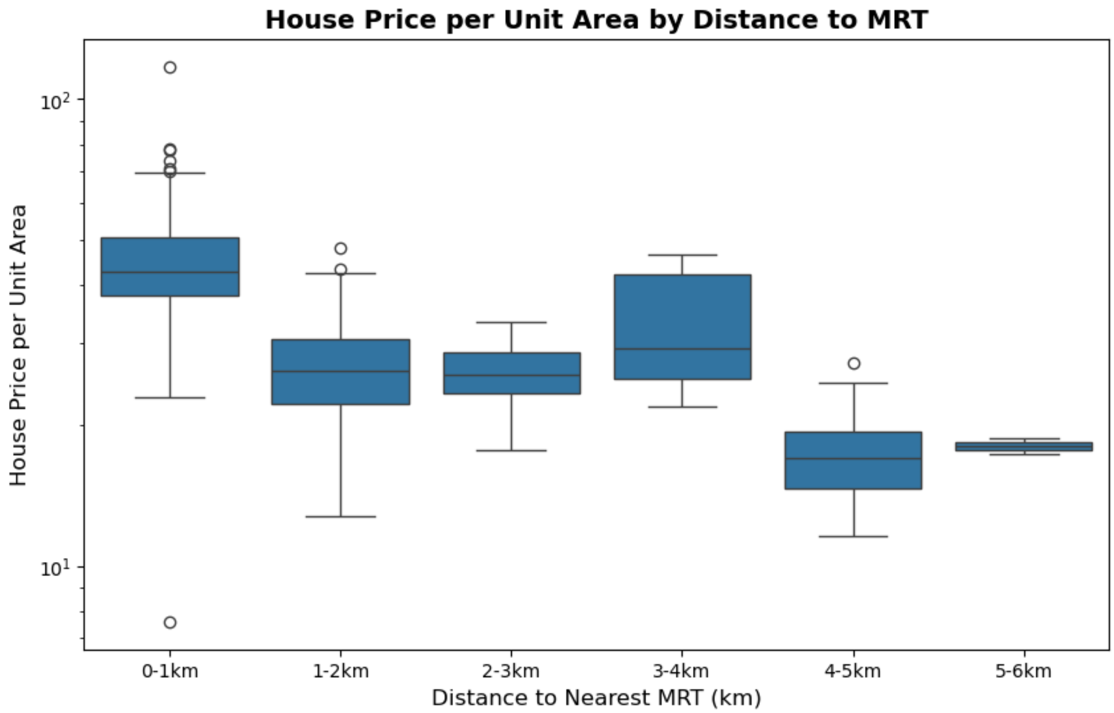
[145]:
```
#Scatterplot Distance to the nearest MRT station vs House price of unit area:␣
↪This shows a relatively strong negative
# relationship between distance and price. Where the distance is less, the␣
↪price of homes appears to increase.
plt.scatter(x=df['Distance to the nearest MRT station'], y=df['House price of␣
↪unit area'])
plt.title('Distance to the Nearest MRT Station vs House Price of Unit Area',␣
↪fontsize=14, fontweight='bold')
plt.xlabel("Distance to the Nearest MRT Station", fontsize=12)
plt.ylabel("House Price of Unit Area", fontsize=12)
plt.show()
```



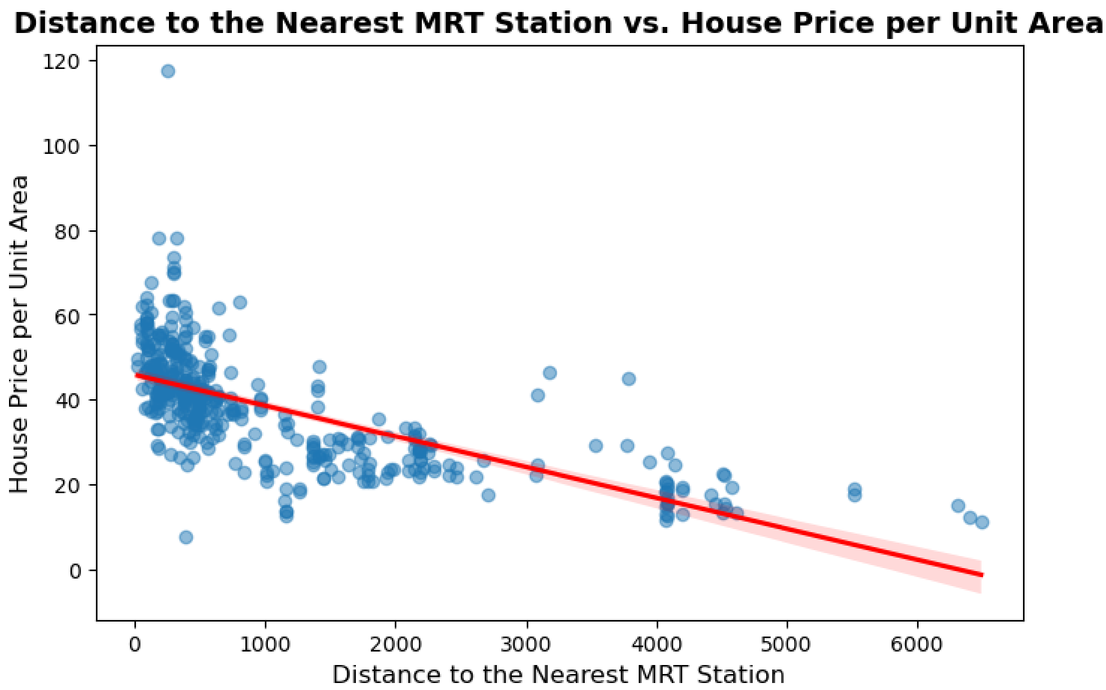
[143]:
```
# Using a multiple boxplot, this compares the price per unit area and the␣
↪distance to MRT: This first iteration is
# challenging to analize because the outliers at the shorter distance are␣
↪compressing the other results.
df['distance_category'] = pd.cut(df['Distance to the nearest MRT station'],
                                 bins=[0, 1000, 2000, 3000, 4000, 5000, 6000],
                                 labels=['0-1km', '1-2km', '2-3km', '3-4km',␣
↪'4-5km', '5-6km'])
plt.figure(figsize=(10, 6))
sns.boxplot(x='distance_category', y='House price of unit area', data=df)
plt.xlabel("Distance to Nearest MRT (km)", fontsize=12)
plt.ylabel("House Price per Unit Area", fontsize=12)
plt.title("House Price per Unit Area by Distance to MRT", fontsize=14,␣
↪fontweight='bold')
plt.show()
```



[141]:
```
# Using a multiple boxplot, this compares the price per unit area and the␣
↪distance to MRT: Tried a log transformation due to high
# outliers compressing the lower values, scaling code provided by ChatGPT. This␣
↪gives a bit more clarity between the distance groups.
# The negative regression pattern is slightly apparent as the distance␣
↪increases.
df['distance_category'] = pd.cut(df['Distance to the nearest MRT station'],
                                 bins=[0, 1000, 2000, 3000, 4000, 5000, 6000],
                                 labels=['0-1km', '1-2km', '2-3km', '3-4km',␣
↪'4-5km', '5-6km'])
plt.figure(figsize=(10, 6))
sns.boxplot(x='distance_category', y='House price of unit area', data=df)
plt.yscale('log')  # Set log scale on y-axis
plt.xlabel("Distance to Nearest MRT (km)", fontsize=12)
plt.ylabel("House Price per Unit Area", fontsize=12)
plt.title("House Price per Unit Area by Distance to MRT", fontsize=14,␣
↪fontweight='bold')
plt.show()
```



[139]:
```
# Regression plot for price and distance to MRT station: Used ChatCPT and given␣
↪python reference notebooks for this plot.
# The negative relationship is clearly defined which agrees with our previous␣
↪analysis. Also the CI is very strong where the
# majority of the results are seen (shorter distance) but becomes less defined␣
↪as the distance increases.
plt.figure(figsize=(8, 5))
sns.regplot(x='Distance to the nearest MRT station', y='House price of unit␣
↪area', data=df, scatter_kws={'alpha':0.5}, line_kws={'color':'red'})
plt.xlabel("Distance to the Nearest MRT Station", fontsize=12)
plt.ylabel("House Price per Unit Area", fontsize=12)
plt.title("Distance to the Nearest MRT Station vs. House Price per Unit Area",␣
↪fontsize=14, fontweight='bold')
plt.show()
```



0.3 Number of Convenience Stores vs House Price of Unit Area: Question 2 #3

[95]:
```
# Obtain the summary statistics on the requested comparison features, Number of␣
↪convenience stores and house price
df[['Number of convenience stores','House price of unit area']].describe()
```

[95]:

	Number of convenience stores	House price of unit area
count	414.000000	414.000000
mean	4.094203	37.980193
std	2.945562	13.606488
min	0.000000	7.600000
25%	1.000000	27.700000
50%	4.000000	38.450000
75%	6.000000	46.600000
max	10.000000	117.500000

[97]:
```
# Compute correlation between number of stores and price: The results show a␣
↪moderate positive correlation between these
# features.
correlation_c = df[['Number of convenience stores', 'House price of unit␣
↪area']].corr()
correlation_c
```

```
[97]:                                   Number of convenience stores  \
      Number of convenience stores                          1.000000
      House price of unit area                              0.571005

                                        House price of unit area
      Number of convenience stores                      0.571005
      House price of unit area                          1.000000
```

```
[101]: # Convenience Stores Skewness: This results shows a very slight right-skew but␣
        ↪nearly normal. There may be a few outliers on
       # the right.
       df["Number of convenience stores"].skew()
```

```
[101]: 0.15460656758377123
```

```
[103]: # Convenience Stores Kurtosis: This shows that there are flatter peaks and␣
        ↪fewer extreme values in this
       # distribution.
       df["Number of convenience stores"].kurt()
```

```
[103]: -1.0657514990134194
```

```
[105]: # House price Skewness: This result is more moderately right-skewed than age,␣
        ↪showing there may be some higher
       # outliers causing the skew.
       df["House price of unit area"].skew()
```

```
[105]: 0.5998525842660576
```

```
[107]: # House price Kurtosis: Because this result is higher than normal, the␣
        ↪distribution is more sharply peaked with
       # quite a few outliers.
       df["House price of unit area"].kurt()
```

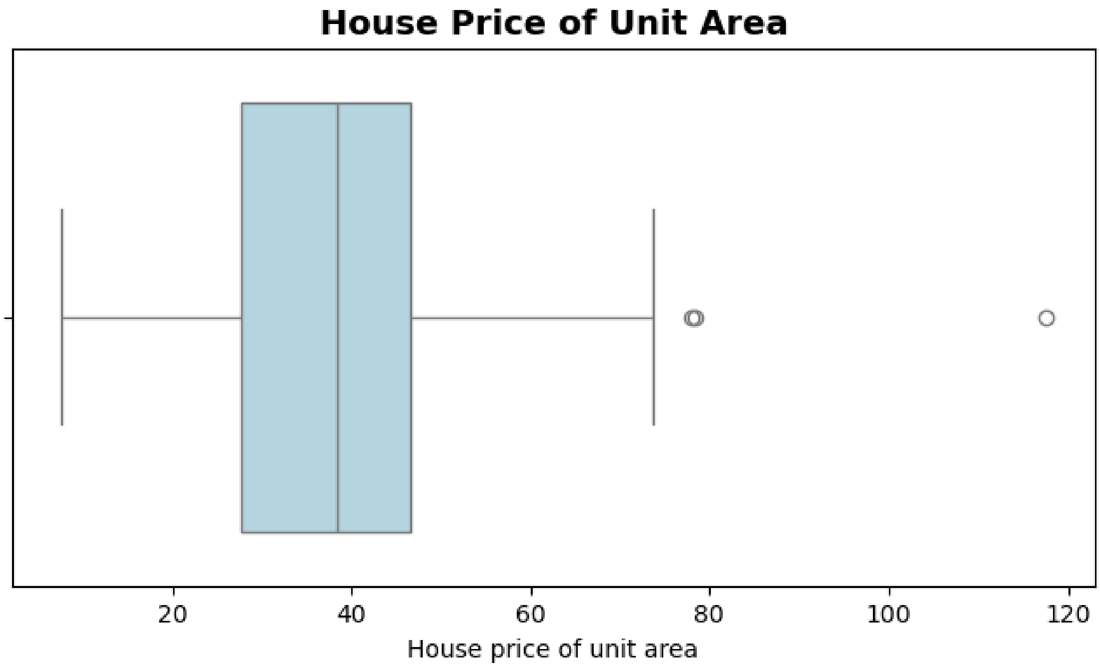
```
[107]: 2.1790970477396163
```

```
[113]: # Used ChatGPT for a code to calculate p value and Pearson's correlation (same␣
        ↪as correlation found above)
       from scipy.stats import pearsonr

       # Calculate Pearson correlation and p-value
       corr_value_c, p_value_c = pearsonr(df['Number of convenience stores'],␣
        ↪df['House price of unit area'])

       print(f"Pearson's correlation: {corr_value_c:.3f}")
       print(f"P-value: {p_value_c:.3f}")
```

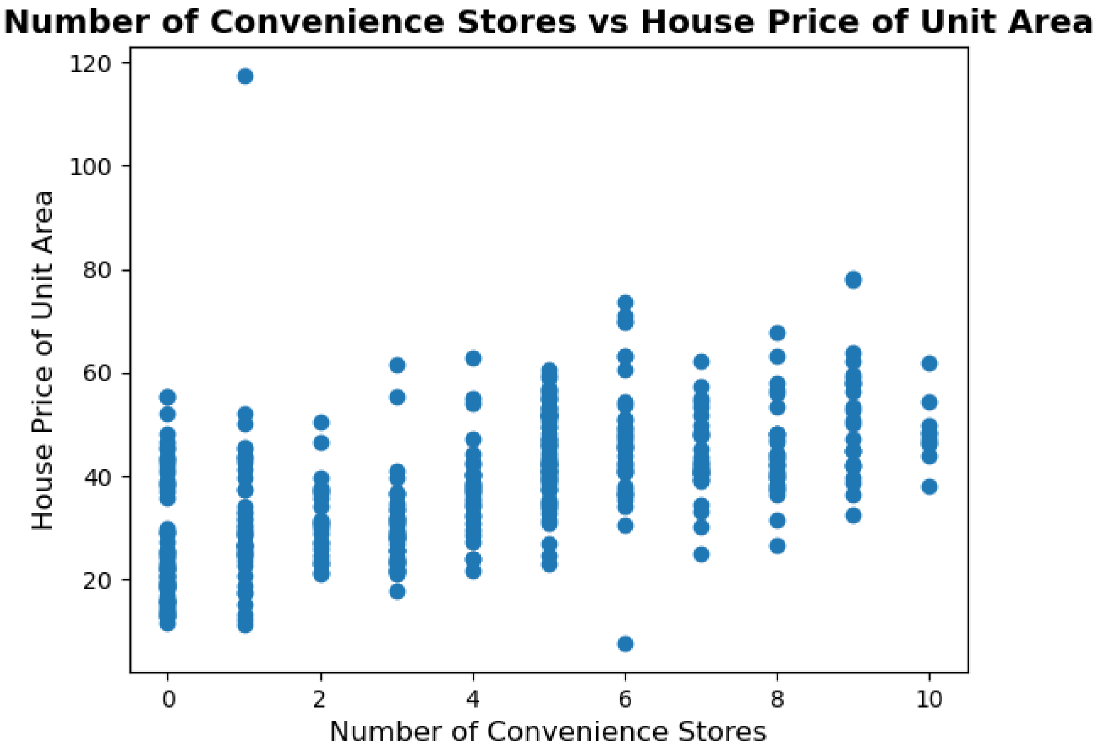
```
Pearson's correlation: 0.571
P-value: 0.000
```

[135]:
```
# Boxplot for Convenience Stores: This graph shows a very slight right-skew
↪which confirms what we saw in the skewness value.
plt.figure(figsize=(8, 4))
sns.boxplot(x=df["Number of convenience stores"], color="lightblue")
plt.title("Number of Convenience Stores", fontsize=14, fontweight='bold')
plt.show()
```

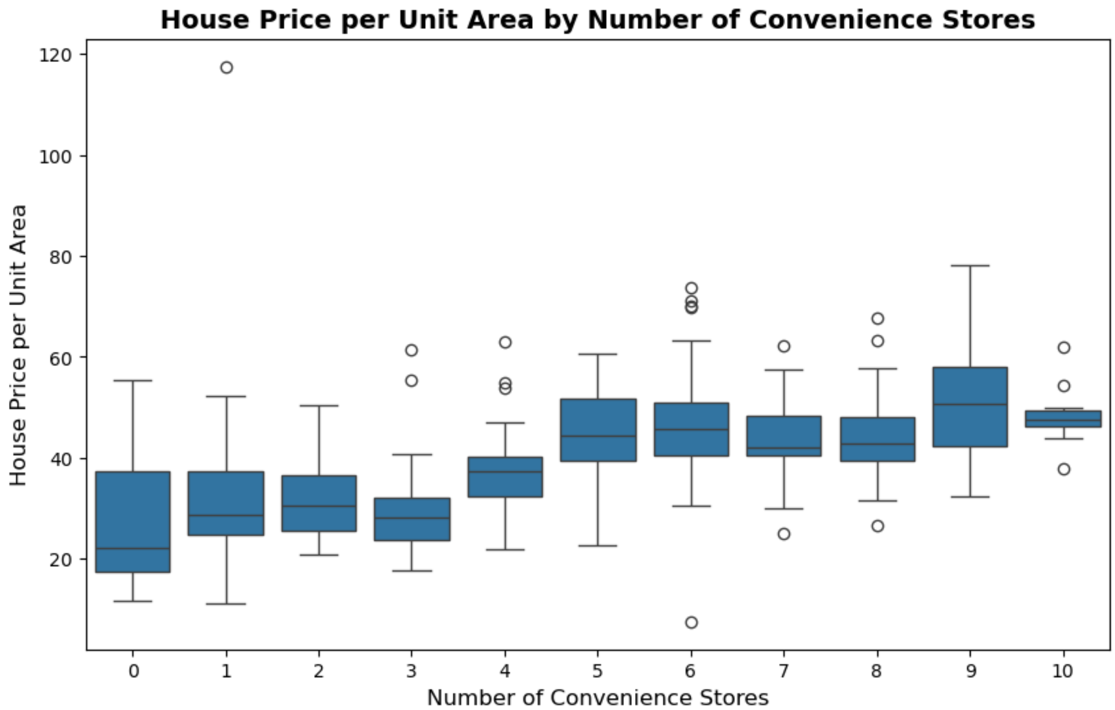
[137]:
```
# Boxplot for Price: This confirms that there is a more defined right-skew with
↪multiple large outliers on the far
# right. This is much more right-skewed than convenience stores.
plt.figure(figsize=(8, 4))
sns.boxplot(x=df["House price of unit area"], color="lightblue")
plt.title("House Price of Unit Area", fontsize=14, fontweight='bold')
plt.show()
```



[131]:
```
#Scatterplot Number of Convenience Stores vs House Price: There is a positive␣
↪relationship between these factors. It
# shows that as the number of stores increases, the price of homes increases as␣
↪well. There are a few outliers but they
# do not have a strong impact on the relationship.
plt.scatter(x=df['Number of convenience stores'], y=df['House price of unit␣
↪area'])
plt.title('Number of Convenience Stores vs House Price of Unit Area',␣
↪fontsize=14, fontweight='bold')
plt.xlabel("Number of Convenience Stores", fontsize=12)
plt.ylabel("House Price of Unit Area", fontsize=12)
plt.show()
```




```
[125]: # Using a multiple boxplot, this compares the price per unit area and␣
       ↪convenience stores: The positive relationship is
       # apparent in this chart. The outliers are also visible within the groups.
       plt.figure(figsize=(10, 6))
       sns.boxplot(x='Number of convenience stores', y='House price of unit area',␣
       ↪data=df)

       plt.xlabel("Number of Convenience Stores", fontsize=12)
       plt.ylabel("House Price per Unit Area", fontsize=12)
       plt.title("House Price per Unit Area by Number of Convenience Stores",␣
       ↪fontsize=14, fontweight='bold')
       plt.show()
```

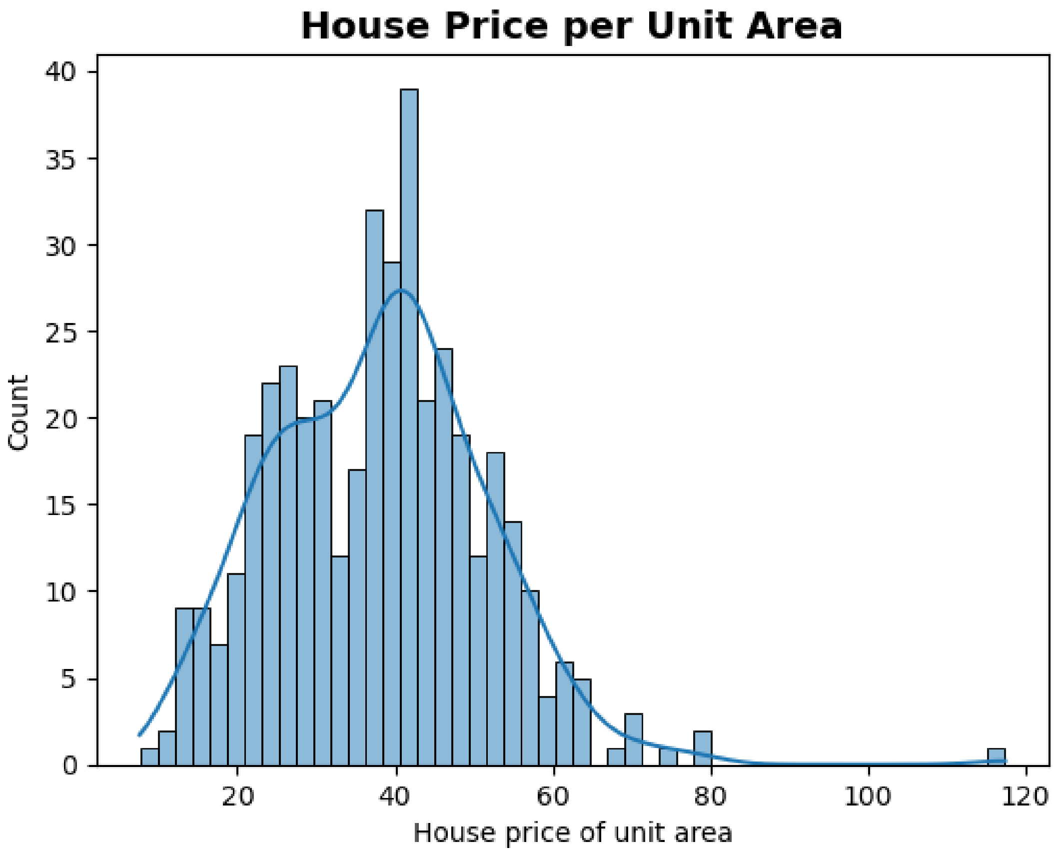


[129]:
```
# Regression plot for price and convenience stores: Used ChatCPT and given␣
↪python reference notebooks for this plot.
# The postive relationship is clearly defined which agrees with our previous␣
↪analysis. Also the CI is very strong within the
# median number of stores where many of the points are located.
plt.figure(figsize=(8, 5))
sns.regplot(x='Number of convenience stores', y='House price of unit area',␣
↪data=df, scatter_kws={'alpha':0.5}, line_kws={'color':'red'})
plt.xlabel("Number of convenience stores", fontsize=12)
plt.ylabel("Price per Unit Area", fontsize=12)
plt.title("Number of convenience stores vs. House Price per Unit Area",␣
↪fontsize=14, fontweight='bold')
plt.show()
```

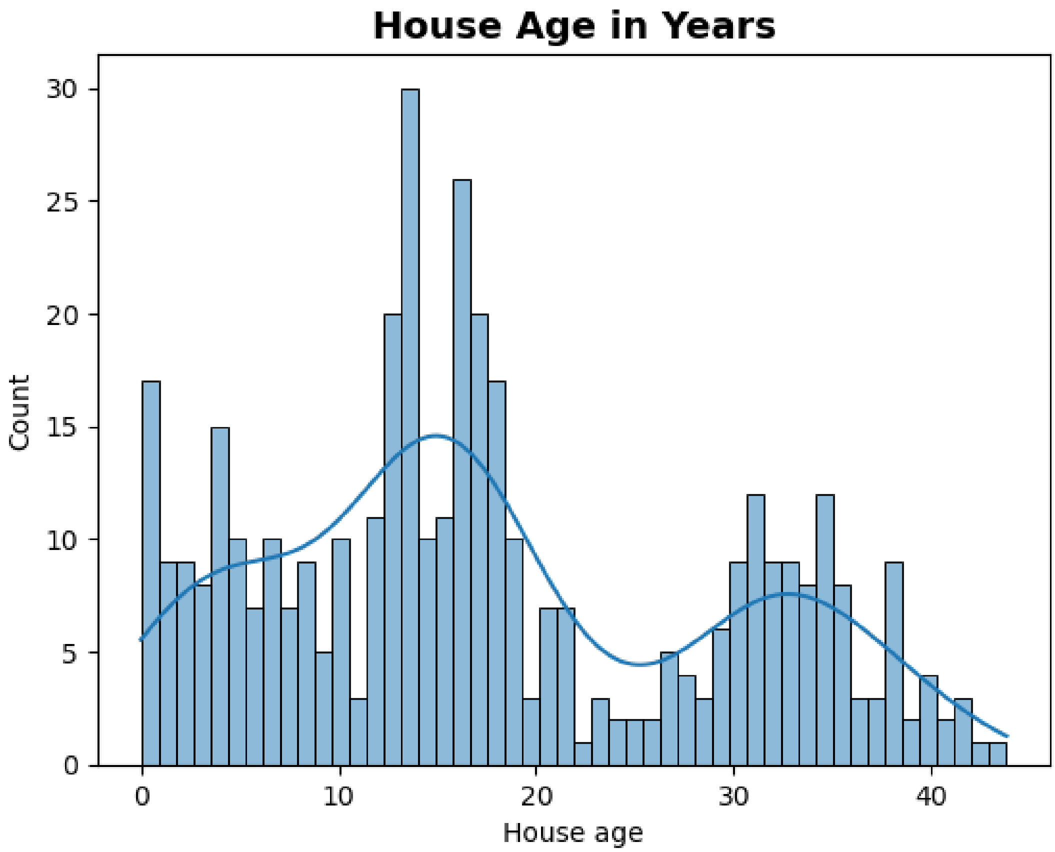
0.4 Simple Regression Model House Price vs Age of House: Question 2 #4

Utilized sample code provided in Lab 6 Simple Linear Regression

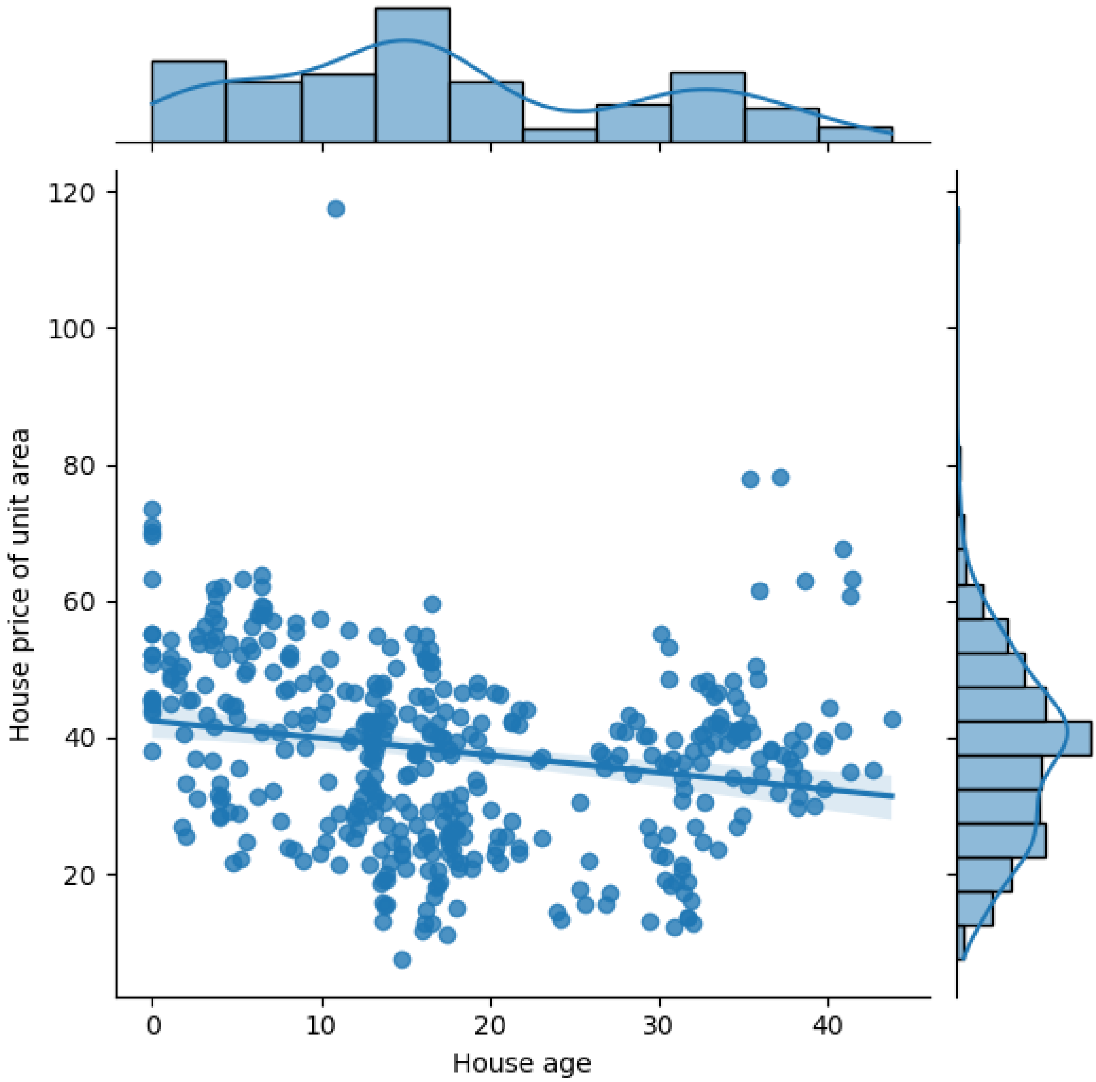
[184]:
```
# Histogram for the house price which is the dependent variable
sns.histplot(data=df, x=df['House price of unit area'], kde=True, bins=50,␣
 ↪element="bars")
plt.title("House Price per Unit Area", fontsize=14, fontweight='bold')
plt.show()
```



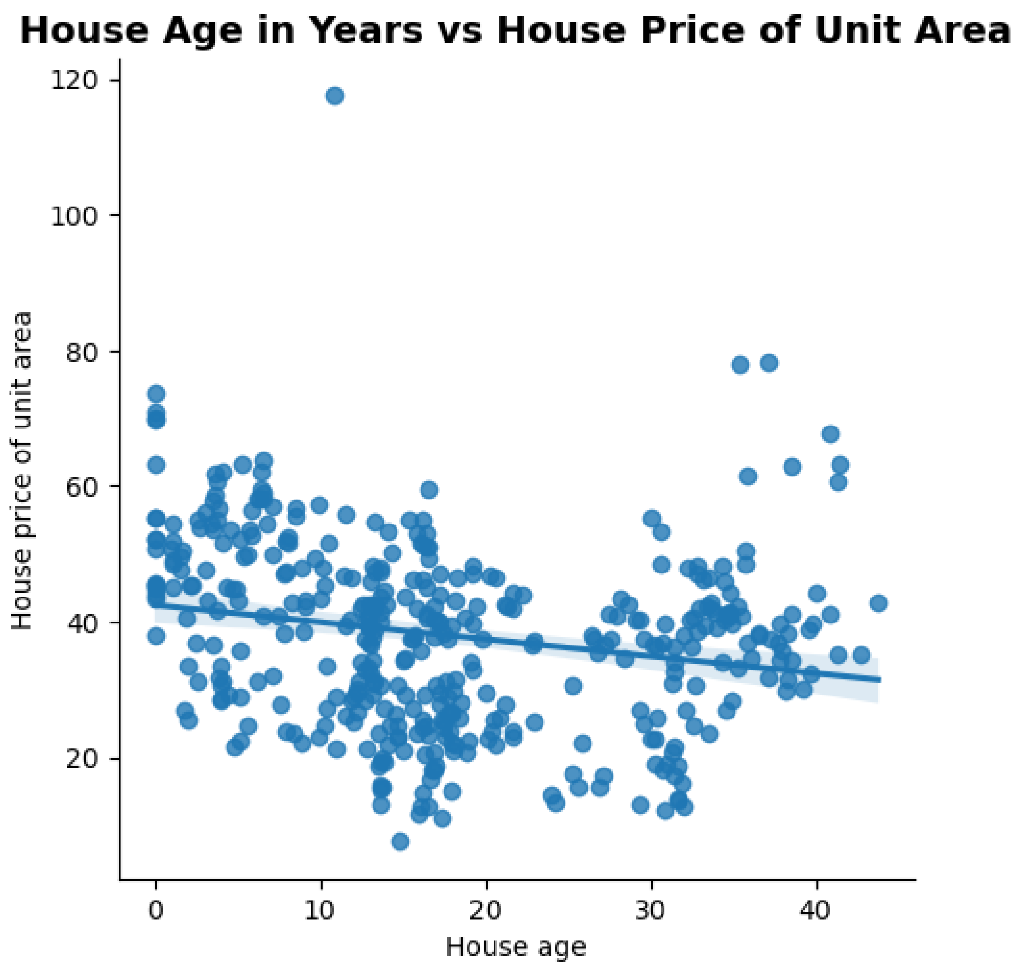
[182]:
```
# Histogram for the house age which is the independent variable
sns.histplot(data=df, x=df['House age'], kde=True, bins=50, element="bars")
plt.title("House Age in Years", fontsize=14, fontweight='bold')
plt.show()
```



[180]:
```
# Scatterplot showing the negative relationship between the two variables
sns.jointplot(x = 'House age', y = 'House price of unit area',data=df,␣
  ↪kind='reg')
plt.show()
```



[178]:
```
# Isolation of the scatterplot for the negative relationship between the two
↪variables.
sns.lmplot(x = 'House age',y = "House price of unit area", data=df)
plt.title("House Age in Years vs House Price of Unit Area", fontsize=14,
↪fontweight='bold')
plt.show()
```




```
[186]: # Define dependent variable
       y = df['House price of unit area']
```

```
[188]: y
```

```
[188]: 0      37.9
       1      42.2
       2      47.3
       3      54.8
       4      43.1
              …
       409    15.4
       410    50.0
       411    40.6
       412    52.5
       413    63.9
       Name: House price of unit area, Length: 414, dtype: float64
```

[190]:
```
# Define independent variable
X = df[['House age']]
```

[192]:
```
X
```

[192]:
```
     House age
0         32.0
1         19.5
2         13.3
3         13.3
4          5.0
..         ...
409       13.7
410        5.6
411       18.8
412        8.1
413        6.5

[414 rows x 1 columns]
```

[200]:
```
# Had to install scikit-Learn to use the model
pip install scikit-learn
```

```
Collecting scikit-learn
  Downloading scikit_learn-1.6.1-cp310-cp310-macosx_10_9_x86_64.whl (12.1 MB)
                           12.1/12.1 MB
3.3 MB/s eta 0:00:0000:0100:01
Requirement already satisfied: numpy>=1.19.5 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.6.0 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from scikit-learn) (1.10.0)
Collecting threadpoolctl>=3.1.0
  Downloading threadpoolctl-3.5.0-py3-none-any.whl (18 kB)
Collecting joblib>=1.2.0
  Downloading joblib-1.4.2-py3-none-any.whl (301 kB)
                           301.8/301.8
kB 7.7 MB/s eta 0:00:0000:01
Installing collected packages: threadpoolctl, joblib, scikit-learn
Successfully installed joblib-1.4.2 scikit-learn-1.6.1 threadpoolctl-3.5.0
Note: you may need to restart the kernel to use updated packages.
```

[202]:
```
from sklearn.model_selection import train_test_split #cross validation, avoid␣
 ↪overfitting
```

[204]:
```python
# Define test and train variables. Assign test size to 30%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=100)
```

[208]:
```python
# Confirm count of results in X_train
print(X_train)
```

```
     House age
287       19.2
31        29.6
209       34.8
239       18.1
47        35.9
..         ...
343       33.5
359        5.6
323       28.6
280        2.3
8         31.7

[289 rows x 1 columns]
```

[210]:
```python
# Confirm count of results in y_train
print(y_train)
```

```
287    32.9
31     25.0
209    40.9
239    29.7
47     61.5
       ...
343    46.6
359    24.7
323    42.5
280    45.4
8      18.8
Name: House price of unit area, Length: 289, dtype: float64
```

[212]:
```python
# Confirm count of results in X_test
print(X_test)
```

```
     House age
121       13.6
353        4.1
96         6.4
43        34.4
125        1.1
..         ...
```

```
248        19.0
84         15.1
409        13.7
80         11.8
161        19.2

[125 rows x 1 columns]
```

[214]:
```
# Confirm count of results in y_test
print(y_test)
```

```
121    48.0
353    31.3
96     59.5
43     34.1
125    48.6
       …
248    22.3
84     43.7
409    15.4
80     40.3
161    39.6
Name: House price of unit area, Length: 125, dtype: float64
```

[216]:
```
# Import linear regression package from sklearn
from sklearn.linear_model import LinearRegression
```

[218]:
```
# Define linear regression variable
LR = LinearRegression()
```

[220]:
```
LR.fit(X_train,y_train) # only use training data to get intercept and slope
```

[220]:
```
LinearRegression()
```

[222]:
```
# Print intercept and slope obtained from training data
print('Regression Intercept:', LR.intercept_)
print('Regression Coefficients:',LR.coef_)
```

```
Regression Intercept: 42.57455942358807
Regression Coefficients: [-0.27024956]
```

[224]:
```
# Define prediction variable for training
LR_Predictions_Train = LR.predict(X_train)
```

[226]:
```
y_train
```

[226]:
```
287    32.9
31     25.0
209    40.9
239    29.7
47     61.5
       …
343    46.6
359    24.7
323    42.5
280    45.4
8      18.8
Name: House price of unit area, Length: 289, dtype: float64
```

[228]:
```
# Review train prediction data
LR_Predictions_Train
```

[228]:
```
array([37.38576781, 34.57517236, 33.16987463, 37.68304233, 32.87260011,
       41.11521178, 40.81793726, 38.98024023, 40.16933831, 37.79114216,
       40.980087  , 40.46661283, 35.38592105, 41.73678578, 38.35866624,
       38.8180905 , 41.601661  , 38.1965165 , 37.98031685, 32.89962506,
       33.16987463, 38.27759137, 41.25033656, 37.89924198, 33.38607428,
       31.73755194, 33.81847358, 42.0340603 , 33.14284967, 41.70976082,
       38.03436676, 31.84565177, 39.43966449, 33.73739871, 31.41325247,
       38.1965165 , 36.7101439 , 37.22361807, 34.65624722, 38.89916537,
       42.16918508, 32.33210098, 36.03452   , 33.22392454, 37.81816711,
       42.57455942, 40.16933831, 38.11544163, 41.14223674, 32.98069993,
       38.00734181, 40.46661283, 32.14292629, 37.00741842, 41.30438648,
       42.57455942, 38.3856912 , 42.57455942, 33.89954845, 34.06169819,
       38.00734181, 38.08841668, 31.81862681, 40.65578753, 41.52058613,
       39.08834006, 36.7101439 , 41.46653621, 38.25056641, 33.57524897,
       38.73701563, 33.65632384, 31.03490308, 42.2772849 , 34.41302262,
       42.06108525, 39.0613151 , 37.84519207, 36.57501912, 37.92626694,
       38.14246659, 40.84496222, 38.57486589, 39.46668945, 39.89908875,
       39.33156467, 38.16949155, 42.2772849 , 38.16949155, 35.73724548,
       40.65578753, 37.79114216, 34.1157481 , 32.30507603, 33.6833488 ,
       40.89901213, 32.8185502 , 38.60189085, 41.27736152, 33.41309923,
       35.19674635, 38.79106554, 38.00734181, 42.57455942, 34.00764827,
       38.60189085, 36.84526869, 38.89916537, 39.14238997, 40.87198718,
       37.16956816, 34.08872314, 39.03429015, 38.46676606, 34.65624722,
       38.89916537, 34.25087288, 31.52135229, 34.44004757, 42.57455942,
       41.49356117, 39.00726519, 38.1965165 , 34.27789784, 33.22392454,
       37.60196746, 41.33141143, 40.73686239, 33.2509495 , 40.14231336,
       38.35866624, 38.89916537, 39.08834006, 42.00703534, 34.1157481 ,
       34.57517236, 37.65601738, 38.27759137, 32.49425072, 41.1692617 ,
       31.87267673, 35.11567148, 38.14246659, 40.52066274, 42.11513517,
       37.03444338, 41.06116187, 38.41271615, 39.35858962, 38.92619032,
       35.25079626, 38.84511545, 41.1692617 , 41.62868595, 38.22354146,
```

```
       41.8448856 , 37.00741842, 42.57455942, 37.87221703, 33.76442367,
       39.87206379, 37.41279277, 40.1152884 , 32.52127568, 40.27743814,
       41.49356117, 39.0613151 , 42.57455942, 41.65571091, 36.35881947,
       39.79098892, 33.71037375, 34.22384792, 40.980087  , 34.30492279,
       39.27751475, 37.65601738, 36.92634355, 34.22384792, 34.06169819,
       38.87214041, 38.27759137, 36.84526869, 37.73709224, 34.35897271,
       38.68296572, 36.41286939, 34.38599766, 41.46653621, 42.57455942,
       31.98077655, 38.6289158 , 42.2772849 , 41.57463604, 38.8180905 ,
       32.87260011, 38.08841668, 39.0613151 , 34.30492279, 40.41256292,
       36.08856991, 39.14238997, 31.52135229, 37.68304233, 41.73678578,
       41.62868595, 38.1965165 , 38.60189085, 33.60227393, 35.30484618,
       38.8180905 , 33.4941741 , 37.38576781, 32.16995124, 37.81816711,
       41.03413691, 40.79091231, 41.49356117, 34.08872314, 36.35881947,
       33.9265734 , 34.46707253, 38.52081598, 37.87221703, 34.35897271,
       38.14246659, 37.79114216, 35.57509574, 37.92626694, 37.06146834,
       33.79144862, 39.00726519, 32.22400116, 30.73762856, 35.14269644,
       40.79091231, 35.43997096, 34.95352174, 32.22400116, 39.11536502,
       33.00772489, 33.52119906, 39.43966449, 41.87191056, 42.57455942,
       38.98024023, 41.19628665, 34.00764827, 37.98031685, 39.65586414,
       38.35866624, 39.95313866, 34.62922227, 37.46684268, 37.49386764,
       40.1152884 , 41.22331161, 37.84519207, 42.57455942, 40.5476877 ,
       41.19628665, 37.62899242, 33.03474985, 38.84511545, 42.57455942,
       42.08811021, 33.95359836, 38.89916537, 34.38599766, 38.98024023,
       33.84549854, 39.35858962, 38.89916537, 40.38553796, 39.60181423,
       32.38615089, 39.00726519, 38.11544163, 42.30430986, 38.98024023,
       39.27751475, 39.00726519, 33.76442367, 37.71006729, 33.52119906,
       41.06116187, 34.84542192, 41.95298543, 34.00764827])
```

[230]:
```python
# Define prediction variable for test
LR_Predictions_Test = LR.predict(X_test)
```

[232]:
```python
LR_Predictions= LR.predict(X)
```

[234]:
```python
# Review test prediction data
print(LR_Predictions)
```

```
[33.9265734  37.30469294 38.98024023 38.98024023 41.22331161 40.65578753
 33.2509495  37.08849329 34.00764827 37.73709224 33.16987463 40.87198718
 39.0613151  37.06146834 39.00726519 32.92665002 42.57455942 37.79114216
 38.00734181 42.16918508 41.35843639 39.73693901 38.60189085 39.84503884
 31.87267673 34.65624722 41.73678578 39.76396397 37.38576781 40.65578753
 35.57509574 34.57517236 32.33210098 38.11544163 38.41271615 38.8180905
 38.60189085 39.33156467 41.73678578 38.1965165  38.89916537 38.03436676
 32.8185502  33.27797445 41.8448856  32.68342541 36.7101439  32.87260011
 36.03452    34.62922227 36.7101439  34.1157481  33.89954845 38.98024023
 38.22354146 34.00764827 33.4941741  41.62868595 34.38599766 38.98024023
 39.60181423 41.14223674 37.92626694 41.87191056 37.84519207 31.73755194
```

42.30430986	40.27743814	34.35897271	39.19643988	40.79091231	32.98069993
33.79144862	38.84511545	40.73686239	39.2504898	32.87260011	37.03444338
32.25102611	37.71006729	39.38561458	34.25087288	39.00726519	35.73724548
38.49379102	42.57455942	42.08811021	38.00734181	40.16933831	36.35881947
42.57455942	40.1152884	37.00741842	33.95359836	31.52135229	40.41256292
40.84496222	34.89947183	38.14246659	40.84496222	37.84519207	39.14238997
42.2772849	42.57455942	33.73739871	42.57455942	37.92626694	39.27751475
34.08872314	41.49356117	40.38553796	33.57524897	39.89908875	38.57486589
34.30492279	37.00741842	34.22384792	38.89916537	35.73724548	38.08841668
38.98024023	38.89916537	34.06169819	42.57455942	39.89908875	42.2772849
32.14292629	41.54761108	31.41325247	32.16995124	34.57517236	41.49356117
35.38592105	37.71006729	33.54822402	37.46684268	39.4937144	38.89916537
39.87206379	39.08834006	38.1965165	41.19628665	37.22361807	38.89916537
39.35858962	42.00703534	42.57455942	41.70976082	38.14246659	33.14284967
32.89962506	41.25033656	39.33156467	40.81793726	38.00734181	38.84511545
34.27789784	38.22354146	39.43966449	38.3856912	41.62868595	37.38576781
38.25056641	40.27743814	42.57455942	38.87214041	42.57455942	34.95352174
35.11567148	40.30446309	36.08856991	41.601661	40.79091231	31.41325247
41.4124863	34.41302262	38.8180905	33.65632384	39.03429015	38.79106554
35.30484618	39.43966449	38.92619032	37.98031685	38.76404058	34.08872314
36.92634355	40.16933831	33.16987463	38.16949155	33.03474985	39.00726519
30.73762856	39.95313866	38.46676606	38.46676606	36.41286939	33.27797445
33.38607428	37.65601738	37.87221703	39.03429015	32.22400116	38.35866624
37.71006729	39.11536502	36.57501912	32.16995124	39.46668945	33.16987463
41.1692617	42.57455942	37.81816711	40.89901213	37.68304233	37.38576781
32.35912594	35.00757166	38.89916537	34.65624722	32.52127568	40.14231336
34.30492279	40.1152884	33.2509495	42.2772849	38.11544163	33.81847358
39.35858962	34.19682297	41.49356117	38.1965165	35.25079626	31.84565177
40.41256292	39.08834006	41.601661	39.0613151	39.11536502	37.68304233
39.60181423	38.87214041	42.0340603	33.71037375	41.27736152	40.5476877
38.14246659	36.7101439	37.43981772	37.71006729	31.98077655	34.00764827
40.980087	34.35897271	42.2772849	34.06169819	38.6289158	37.89924198
42.57455942	37.79114216	37.98031685	38.1965165	38.27759137	41.52058613
33.76442367	38.33164128	37.7641172	33.19689958	37.92626694	37.81816711
39.65586414	37.79114216	39.0613151	39.00726519	35.14269644	42.16918508
37.41279277	36.84526869	42.57455942	41.87191056	41.95298543	41.30438648
42.0340603	33.52119906	38.52081598	34.44004757	40.980087	37.38576781
38.08841668	38.8180905	32.38615089	41.65571091	37.84519207	39.16941493
35.43997096	37.65601738	39.19643988	33.14284967	38.06139172	33.60227393
41.89893552	32.30507603	38.11544163	32.22400116	37.16956816	38.1965165
38.68296572	39.79098892	38.14246659	34.38599766	38.14246659	36.81824373
33.00772489	40.33148805	41.57463604	38.35866624	38.98024023	38.35866624
40.65578753	33.22392454	38.92619032	38.00734181	39.08834006	34.84542192
39.22346484	32.68342541	41.46653621	41.62868595	38.27759137	38.89916537
33.9265734	35.65617061	31.81862681	40.46661283	34.46707253	35.19674635
41.19628665	34.1157481	34.06169819	42.11513517	33.4941741	39.0613151
41.03413691	33.52119906	33.22392454	42.57455942	39.00726519	37.87221703
41.33141143	40.46661283	39.00726519	41.49356117	37.60196746	41.46653621

```
 39.27751475 41.54761108 39.79098892 42.57455942 42.2772849  41.06116187
 33.6833488  31.38622751 37.95329189 33.84549854 33.03474985 37.89924198
 38.73701563 38.52081598 37.65601738 37.11551825 38.27759137 41.46653621
 33.41309923 42.57455942 41.11521178 36.7101439  38.60189085 41.52058613
 32.49425072 42.57455942 38.76404058 40.41256292 38.16949155 34.71029714
 38.22354146 37.62899242 42.57455942 38.1965165  39.76396397 31.52135229
 33.71037375 40.89901213 31.03490308 38.00734181 33.76442367 36.84526869
 32.54830063 39.03429015 38.60189085 39.14238997 35.33187113 40.52066274
 39.14238997 34.22384792 38.14246659 36.35881947 42.06108525 41.1692617
 37.57494251 38.87214041 41.06116187 37.49386764 40.38553796 40.81793726]
```

[236]:

```
# Capture R^2 values for test and training
print('R^2 for training set:', LR.score(X_train, y_train))
print('R^2 for test set:', LR.score(X_test, y_test))
```

```
R^2 for training set: 0.04799221706333745
R^2 for test set: 0.024370363655339844
```

[238]:

```
# import dependencies
from sklearn import metrics
```

[240]:

```
# Capture Root Mean Squared Error
RMSE_train = np.sqrt(metrics.mean_squared_error(y_train, LR_Predictions_Train))
RMSE_test = np.sqrt(metrics.mean_squared_error(y_test, LR_Predictions_Test))
print('RMSE for training set:', RMSE_train)
print('RMSE for test set:', RMSE_test) # training and test errors similar
```

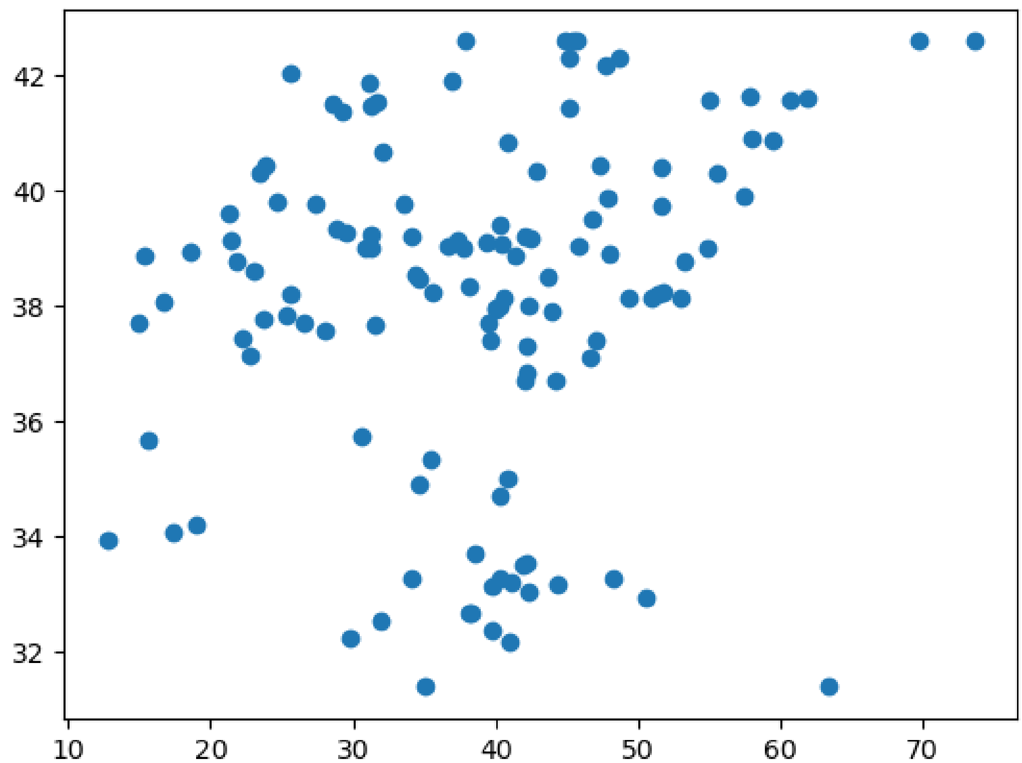
```
RMSE for training set: 13.921708475703896
RMSE for test set: 11.693813593030535
```

[244]:

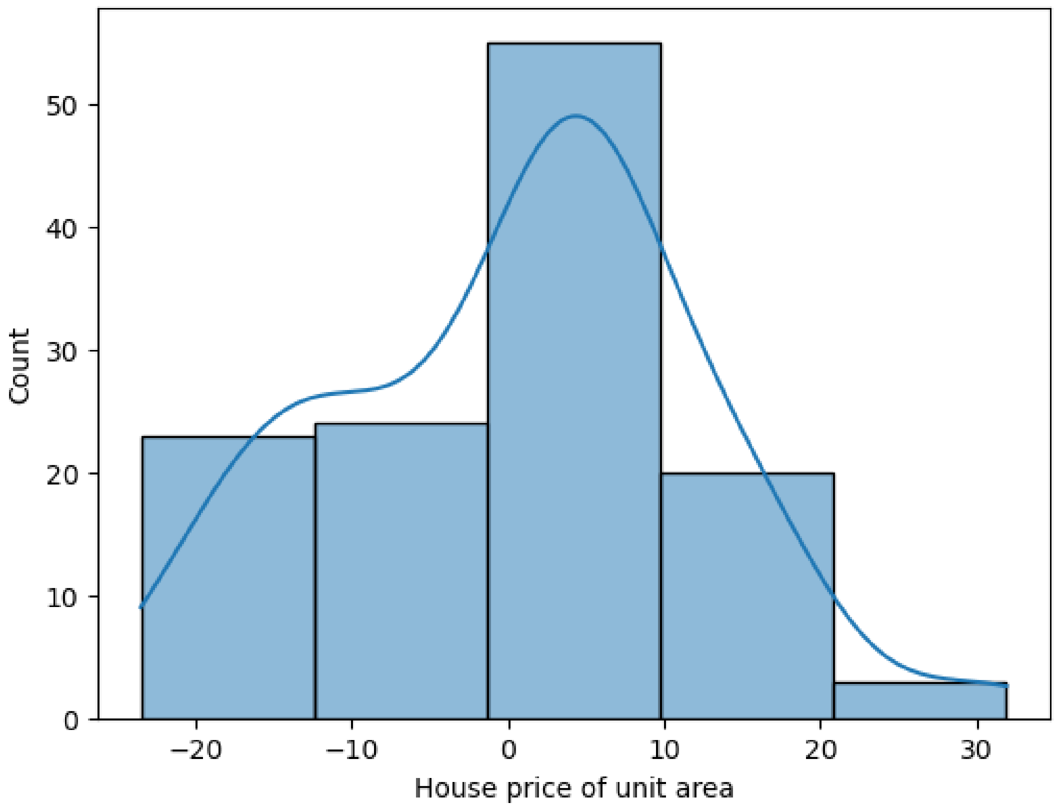
```
# Scatterplot for test data
plt.scatter(y_test,LR_Predictions_Test)
#plt.show()
```

[244]:

```
<matplotlib.collections.PathCollection at 0x7fa150064b80>
```



[250]:
```
# Histogram for test data
sns.histplot(data=df, x=(y_test-LR_Predictions_Test), kde=True, bins=5,␣
 ↪element="bars") # not normal dist
plt.show()
```



0.5 Regression Function: Question 2 #5

0.5.1 The best fitting line is Y = 42.58 - 0.27*X #### Regression Intercept: 42.57455942358807 #### Regression Coefficient (slope): [-0.27024956]

0.6 Regression Model Comparison: Question 2 #10

0.6.1 Single linear regression model results

[263]:
```
# Asked ChatGPT to help with code to make a result similar to .summary() to
↪capture OLS Regression Results for train and test sets

import statsmodels.api as sm
from sklearn.metrics import mean_squared_error, r2_score

# Compute residuals
residuals_train = y_train - LR_Predictions_Train
residuals_test = y_test - LR_Predictions_Test

# R-squared
r2_train = r2_score(y_train, LR_Predictions_Train)
r2_test = r2_score(y_test, LR_Predictions_Test)
```

```python
# Adjusted R-squared
n_train, k = X_train.shape
n_test = X_test.shape[0]

adj_r2_train = 1 - (1 - r2_train) * ((n_train - 1) / (n_train - k - 1))
adj_r2_test = 1 - (1 - r2_test) * ((n_test - 1) / (n_test - k - 1))

# Mean Squared Error & RMSE
mse_train = mean_squared_error(y_train, LR_Predictions_Train)
mse_test = mean_squared_error(y_test, LR_Predictions_Test)

rmse_train = np.sqrt(mse_train)
rmse_test = np.sqrt(mse_test)

# Add constant for statsmodels OLS summary
X_train_with_const = sm.add_constant(X_train)

# Run statsmodels OLS to get summary (for Training Data)
ols_model = sm.OLS(y_train, X_train_with_const).fit()
summary_table = ols_model.summary()

# Print key metrics for training & test sets
print("Scikit-learn Linear Regression Summary:")
print(f"Intercept: {LR.intercept_:.4f}")
print("Coefficients:")
print(pd.Series(LR.coef_, index=X_train.columns))

print("Training Set Performance:")
print(f"R-squared: {r2_train:.4f}")
print(f"Adjusted R-squared: {adj_r2_train:.4f}")
print(f"Mean Squared Error (MSE): {mse_train:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse1_train:.4f}")

print("Test Set Performance:")
print(f"R-squared: {r2_test:.4f}")
print(f"Adjusted R-squared: {adj_r2_test:.4f}")
print(f"Mean Squared Error (MSE): {mse_test:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse1_test:.4f}")

# Display full statsmodels-style summary
print("Statsmodels OLS Summary (Training Data):")
print()
print(summary_table)
```

```
Scikit-learn Linear Regression Summary:
Intercept: 42.5746
```

```
Coefficients:
House age   -0.27025
dtype: float64
Training Set Performance:
R-squared: 0.0480
Adjusted R-squared: 0.0447
Mean Squared Error (MSE): 193.8140
Root Mean Squared Error (RMSE): 13.9217
Test Set Performance:
R-squared: 0.0244
Adjusted R-squared: 0.0164
Mean Squared Error (MSE): 136.7453
Root Mean Squared Error (RMSE): 11.6938
Statsmodels OLS Summary (Training Data):

                               OLS Regression Results
================================================================================
====
Dep. Variable:     House price of unit area   R-squared:
0.048
Model:                                  OLS   Adj. R-squared:
0.045
Method:                       Least Squares   F-statistic:
14.47
Date:                      Sun, 02 Mar 2025   Prob (F-statistic):
0.000174
Time:                              08:09:11   Log-Likelihood:
-1171.1
No. Observations:                       289   AIC:
2346.
Df Residuals:                           287   BIC:
2354.
Df Model:                                 1
Covariance Type:                  nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         42.5746      1.534     27.757      0.000      39.556      45.593
House age     -0.2702      0.071     -3.804      0.000      -0.410      -0.130
==============================================================================
Omnibus:                       47.008   Durbin-Watson:                   2.012
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              122.957
Skew:                           0.747   Prob(JB):                     2.00e-27
Kurtosis:                       5.825   Cond. No.                         40.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
```

specified.

[267]:

```
# Obtain P-value and T statistic for question 2 # 6
from scipy import stats

# Given values from the regression summary (example for House age)
beta_1 = -0.2702  # Slope coefficient
std_err = 0.071  # Standard error of slope
t_statistic = beta_1 / std_err  # Compute t-statistic
df = len(X_train) - 2  # Degrees of freedom (n - k - 1, where k=1 for simple␣
↪regression)

# Compute p-value
p_value = 2 * (1 - stats.t.cdf(abs(t_statistic), df))

print(f"T-statistic: {t_statistic:.3f}")
print(f"P-value: {p_value:.6f}")
```

```
T-statistic: -3.806
P-value: 0.000173
```

[277]:

```
# Based on the previous lecture for 95% is +/- 1.968
from scipy.stats import t

# Degrees of freedom (n - 2)
df = len(X_train) - 2

# Find the critical t-value for a two-tailed test at alpha = 0.05
t_critical = t.ppf(1 - 0.025, df)

print(f"Critical t-value: ±{t_critical:.3f}")
```

```
Critical t-value: ±1.968
```

[ ]:

hw_3_question 2_pt2

March 3, 2025

[2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[4]:
```python
# Read in the data
df = pd.read_csv('/Users/helenamabey/Stats_Spring_2025/Real_estate.csv')
```

[6]:
```python
df.head()
```

[6]:
```
   No  Transaction date  House age  Distance to the nearest MRT station  \
0   1          2012.917       32.0                             84.87882
1   2          2012.917       19.5                            306.59470
2   3          2013.583       13.3                            561.98450
3   4          2013.500       13.3                            561.98450
4   5          2012.833        5.0                            390.56840

   Number of convenience stores  Latitude  Longitude  House price of unit area
0                            10  24.98298  121.54024                      37.9
1                             9  24.98034  121.53951                      42.2
2                             5  24.98746  121.54391                      47.3
3                             5  24.98746  121.54391                      54.8
4                             5  24.97937  121.54245                      43.1
```

[8]:
```python
# Used ChatGPT to help update the given date format to a usable date format.␣
↪Defined a function to capture a standard date as described
# in the initial data definition table in homework.
from datetime import datetime, timedelta

def decimal_year_to_date(decimal_year):
    year = int(decimal_year)
    remainder = decimal_year - year
    start_of_year = datetime(year, 1, 1)
    days_in_year = (datetime(year + 1, 1, 1) - start_of_year).days
    actual_date = start_of_year + timedelta(days=remainder * days_in_year)
    return actual_date.strftime("%Y-%m-%d")
```

```
df['Transaction date'] = [decimal_year_to_date(d) for d in df['Transaction␣
↪date']]

df.head()
```

[8]:
```
   No Transaction date  House age  Distance to the nearest MRT station  \
0   1       2012-12-01       32.0                             84.87882
1   2       2012-12-01       19.5                            306.59470
2   3       2013-08-01       13.3                            561.98450
3   4       2013-07-02       13.3                            561.98450
4   5       2012-10-31        5.0                            390.56840

   Number of convenience stores  Latitude  Longitude  House price of unit area
0                            10  24.98298  121.54024                      37.9
1                             9  24.98034  121.53951                      42.2
2                             5  24.98746  121.54391                      47.3
3                             5  24.98746  121.54391                      54.8
4                             5  24.97937  121.54245                      43.1
```

[10]:
```
# Correct the Transaction date data type
df['Transaction date'] = pd.to_datetime(df['Transaction date'])
df.head()
```

[10]:
```
   No Transaction date  House age  Distance to the nearest MRT station  \
0   1       2012-12-01       32.0                             84.87882
1   2       2012-12-01       19.5                            306.59470
2   3       2013-08-01       13.3                            561.98450
3   4       2013-07-02       13.3                            561.98450
4   5       2012-10-31        5.0                            390.56840

   Number of convenience stores  Latitude  Longitude  House price of unit area
0                            10  24.98298  121.54024                      37.9
1                             9  24.98034  121.53951                      42.2
2                             5  24.98746  121.54391                      47.3
3                             5  24.98746  121.54391                      54.8
4                             5  24.97937  121.54245                      43.1
```

[12]:
```
# Confirmed the data type has been updated
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   No                                   414 non-null    int64
 1   Transaction date                     414 non-null    datetime64[ns]
```

```
 2   House age                               414 non-null    float64
 3   Distance to the nearest MRT station     414 non-null    float64
 4   Number of convenience stores            414 non-null    int64
 5   Latitude                                414 non-null    float64
 6   Longitude                               414 non-null    float64
 7   House price of unit area                414 non-null    float64
dtypes: datetime64[ns](1), float64(5), int64(2)
memory usage: 26.0 KB
```

[14]:
```
# Obtain the summary statistics on the full data set
df.describe()
```

[14]:
```
               No   House age  Distance to the nearest MRT station  \
count  414.000000  414.000000                           414.000000
mean   207.500000   17.712560                          1083.885689
std    119.655756   11.392485                          1262.109595
min      1.000000    0.000000                            23.382840
25%    104.250000    9.025000                           289.324800
50%    207.500000   16.100000                           492.231300
75%    310.750000   28.150000                          1454.279000
max    414.000000   43.800000                          6488.021000

       Number of convenience stores    Latitude   Longitude  \
count                    414.000000  414.000000  414.000000
mean                       4.094203   24.969030  121.533361
std                        2.945562    0.012410    0.015347
min                        0.000000   24.932070  121.473530
25%                        1.000000   24.963000  121.528085
50%                        4.000000   24.971100  121.538630
75%                        6.000000   24.977455  121.543305
max                       10.000000   25.014590  121.566270

       House price of unit area
count                414.000000
mean                  37.980193
std                   13.606488
min                    7.600000
25%                   27.700000
50%                   38.450000
75%                   46.600000
max                  117.500000
```

[17]:
```
# Obtain the summary statistics on the requested comparison features, House age␣
 ↪and House price of unit area
df[['House age','House price of unit area']].describe()
```

[17]:

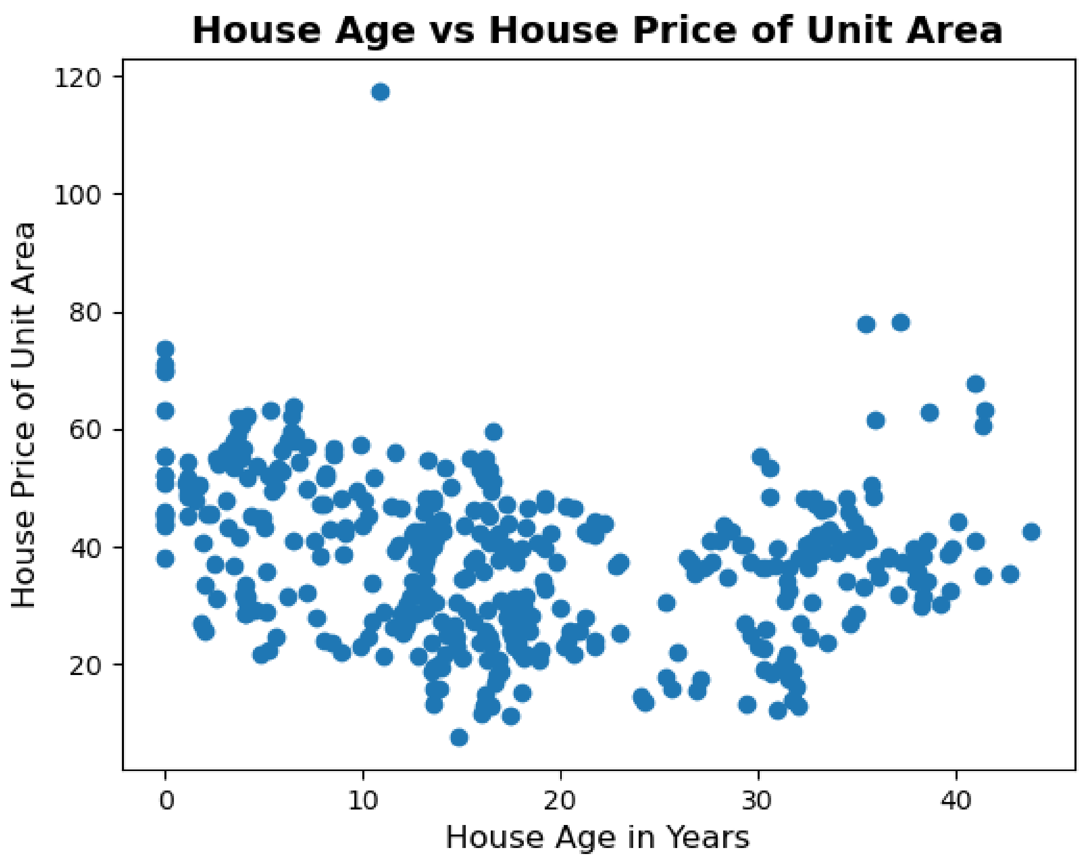
	House age	House price of unit area
count	414.000000	414.000000
mean	17.712560	37.980193
std	11.392485	13.606488
min	0.000000	7.600000
25%	9.025000	27.700000
50%	16.100000	38.450000
75%	28.150000	46.600000
max	43.800000	117.500000

[19]:
```
# Compute correlation between age and price: This shows that while there is a␣
↪negative correlation between the two
# features, it is very small. House prices do fall as the age of a house␣
↪increases but it is not a strong factor.
correlation = df[['House age', 'House price of unit area']].corr()
correlation
```

[19]:

	House age	House price of unit area
House age	1.000000	-0.210567
House price of unit area	-0.210567	1.000000

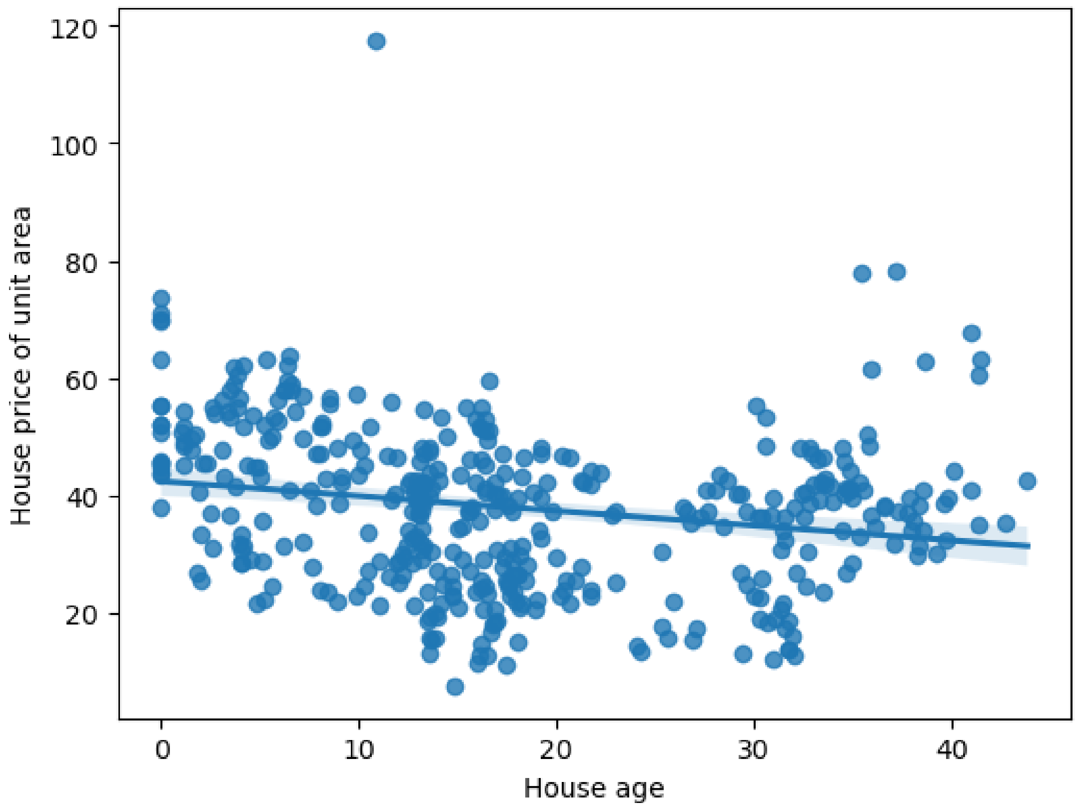
[21]:
```
#Scatterplot House age vs House price of unit area: This shows that there may␣
↪not be a strong relationship to age
# and price. There are a few outliers but generally the price results are␣
↪similar regardless of house age.
plt.scatter(x=df['House age'], y=df['House price of unit area'])
plt.title('House Age vs House Price of Unit Area', fontsize=14,␣
↪fontweight='bold')
plt.xlabel("House Age in Years", fontsize=12)
plt.ylabel("House Price of Unit Area", fontsize=12)
plt.show()
```



[25]:
```
# import dependencies
import seaborn as sns
```

[26]:
```
# Regression plot house age and price: This shows the slight negative␣
↪relationship between
# these two factors. This confirms the correlation results we previously␣
↪reviewed that age has a minimal impact on price.
sns.regplot(data=df, x='House age', y='House price of unit area')
```

[26]:
```
<Axes: xlabel='House age', ylabel='House price of unit area'>
```



[29]:
```
# Import dependencies for linear regression model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# y is vector and X is matrix, if mult you add other characteristics

# Define the feature and target
X = df[['House age']]
y = df['House price of unit area']

model = LinearRegression()
reg=model.fit(X, y)
```

[31]:
```
# Display the model coefficients first is slope, second is intercept
model.coef_, model.intercept_
```

[31]:
```
(array([-0.25148842]), 42.4346970462629)
```

[35]:
```
# Install statsmodels to use packages
pip install statsmodels
```

```
Collecting statsmodels
```

```
  Downloading statsmodels-0.14.4-cp310-cp310-macosx_10_9_x86_64.whl (10.2 MB)
                           10.2/10.2 MB
5.3 MB/s eta 0:00:0000:010:01
Collecting patsy>=0.5.6
  Downloading patsy-1.0.1-py2.py3-none-any.whl (232 kB)
                           232.9/232.9

kB 6.8 MB/s eta 0:00:00
Requirement already satisfied: pandas!=2.1.0,>=1.4 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from statsmodels) (1.5.3)
Requirement already satisfied: packaging>=21.3 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from statsmodels) (23.0)
Requirement already satisfied: numpy<3,>=1.22.3 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from statsmodels) (1.23.5)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from statsmodels) (1.10.0)
Requirement already satisfied: python-dateutil>=2.8.1 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from pandas!=2.1.0,>=1.4->statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from pandas!=2.1.0,>=1.4->statsmodels) (2022.7)
Requirement already satisfied: six>=1.5 in
/Users/helenamabey/opt/anaconda3/envs/PythonData/lib/python3.10/site-packages
(from python-dateutil>=2.8.1->pandas!=2.1.0,>=1.4->statsmodels) (1.16.0)
Installing collected packages: patsy, statsmodels
Successfully installed patsy-1.0.1 statsmodels-0.14.4
Note: you may need to restart the kernel to use updated packages.
```

[37]:

```
# Import dependencies
import statsmodels.api as sm
from statsmodels.stats.anova import anova_lm
import scipy.stats as stats
```

[39]:

```
# Define the features: house age and targetand house price
X = df[['House age']]
y = df['House price of unit area']

# Add a constant (for intercept in the regression model)
X_with_const = sm.add_constant(X)

# Create the linear regression model using statsmodels
# ordinaryleast squares.
model_sm = sm.OLS(y, X_with_const).fit()
```

[41]:
```python
# Print Model Summary
model_summary = model_sm.summary()
print(model_summary)
```

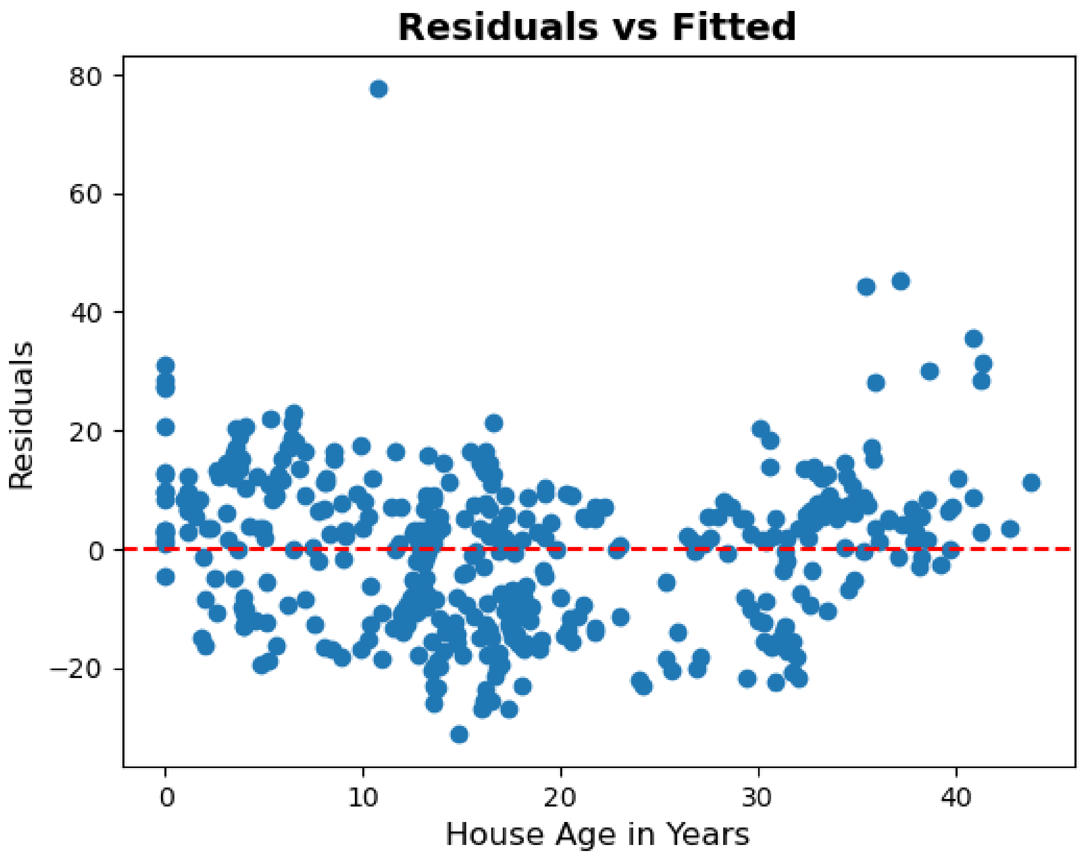
```
                            OLS Regression Results
================================================================================
====
Dep. Variable:     House price of unit area   R-squared:
0.044
Model:                                  OLS   Adj. R-squared:
0.042
Method:                       Least Squares   F-statistic:
19.11
Date:                      Sun, 02 Mar 2025   Prob (F-statistic):
1.56e-05
Time:                              07:45:37   Log-Likelihood:
-1658.3
No. Observations:                       414   AIC:
3321.
Df Residuals:                           412   BIC:
3329.
Df Model:                                 1
Covariance Type:                  nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         42.4347      1.211     35.042      0.000      40.054      44.815
House age     -0.2515      0.058     -4.372      0.000      -0.365      -0.138
==============================================================================
Omnibus:                       48.404   Durbin-Watson:                   1.957
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              119.054
Skew:                           0.589   Prob(JB):                     1.40e-26
Kurtosis:                       5.348   Cond. No.                         39.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

0.1 Regression function for House Age vs House Price of Unit area (Y=a+bX)

The best fitting line is Y = 42.435 - 0.25*X based on this model

[44]:
```python
# Graph the residuals (errors)
residuals = model_sm.resid

# Plot residuals vs. fitted values
plt.scatter(df['House age'], residuals)
```

```
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('House Age in Years', fontsize=12)
plt.ylabel('Residuals', fontsize=12)
plt.title('Residuals vs Fitted', fontsize=14, fontweight='bold')
plt.show()
```



hw_3_ques2_9

March 3, 2025

[1]:
```python
# Import all dependencies for multiple linear regression modeling. Full sample␣
↪code from ChatGPT
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

[3]:
```python
# Read in the data
df = pd.read_csv('/Users/helenamabey/Stats_Spring_2025/Real_estate.csv')
df.head()
```

[3]:
```
   No  Transaction date  House age  Distance to the nearest MRT station  \
0   1          2012.917       32.0                             84.87882
1   2          2012.917       19.5                            306.59470
2   3          2013.583       13.3                            561.98450
3   4          2013.500       13.3                            561.98450
4   5          2012.833        5.0                            390.56840

   Number of convenience stores  Latitude  Longitude  House price of unit area
0                            10  24.98298  121.54024                      37.9
1                             9  24.98034  121.53951                      42.2
2                             5  24.98746  121.54391                      47.3
3                             5  24.98746  121.54391                      54.8
4                             5  24.97937  121.54245                      43.1
```

[5]:
```python
# Update date format
from datetime import datetime, timedelta

def decimal_year_to_date(decimal_year):
    year = int(decimal_year)
    remainder = decimal_year - year
    start_of_year = datetime(year, 1, 1)
    days_in_year = (datetime(year + 1, 1, 1) - start_of_year).days
```

```python
    actual_date = start_of_year + timedelta(days=remainder * days_in_year)
    return actual_date.strftime("%Y-%m-%d")

df['Transaction date'] = [decimal_year_to_date(d) for d in df['Transaction␣
 ↪date']]

df.head()
```

[5]:

```
   No Transaction date  House age  Distance to the nearest MRT station  \
0   1       2012-12-01       32.0                             84.87882
1   2       2012-12-01       19.5                            306.59470
2   3       2013-08-01       13.3                            561.98450
3   4       2013-07-02       13.3                            561.98450
4   5       2012-10-31        5.0                            390.56840

   Number of convenience stores  Latitude  Longitude  House price of unit area
0                            10  24.98298  121.54024                      37.9
1                             9  24.98034  121.53951                      42.2
2                             5  24.98746  121.54391                      47.3
3                             5  24.98746  121.54391                      54.8
4                             5  24.97937  121.54245                      43.1
```

[7]:

```python
# Correct date data type
df['Transaction date'] = pd.to_datetime(df['Transaction date'])
df.head()
```

[7]:

```
   No Transaction date  House age  Distance to the nearest MRT station  \
0   1       2012-12-01       32.0                             84.87882
1   2       2012-12-01       19.5                            306.59470
2   3       2013-08-01       13.3                            561.98450
3   4       2013-07-02       13.3                            561.98450
4   5       2012-10-31        5.0                            390.56840

   Number of convenience stores  Latitude  Longitude  House price of unit area
0                            10  24.98298  121.54024                      37.9
1                             9  24.98034  121.53951                      42.2
2                             5  24.98746  121.54391                      47.3
3                             5  24.98746  121.54391                      54.8
4                             5  24.97937  121.54245                      43.1
```

[9]:

```python
# Obtain the summary statistics on the full data set
df.describe()
```

[9]:

```
               No   House age  Distance to the nearest MRT station  \
count  414.000000  414.000000                           414.000000
mean   207.500000   17.712560                          1083.885689
std    119.655756   11.392485                          1262.109595
```

```
min      1.000000    0.000000                                  23.382840
25%    104.250000    9.025000                                 289.324800
50%    207.500000   16.100000                                 492.231300
75%    310.750000   28.150000                                1454.279000
max    414.000000   43.800000                                6488.021000

       Number of convenience stores    Latitude   Longitude  \
count                    414.000000  414.000000  414.000000
mean                       4.094203   24.969030  121.533361
std                        2.945562    0.012410    0.015347
min                        0.000000   24.932070  121.473530
25%                        1.000000   24.963000  121.528085
50%                        4.000000   24.971100  121.538630
75%                        6.000000   24.977455  121.543305
max                       10.000000   25.014590  121.566270

       House price of unit area
count                414.000000
mean                  37.980193
std                   13.606488
min                    7.600000
25%                   27.700000
50%                   38.450000
75%                   46.600000
max                  117.500000
```

0.1 Multiple Linear Regression Model: Question 2 #9

[11]:
```python
# Obtain the summary statistics on the requested comparison features, House␣
↪age, distance, and House price of unit area
df[['House age','House price of unit area','Distance to the nearest MRT␣
↪station']].describe()
```

[11]:
```
         House age  House price of unit area  \
count  414.000000                414.000000
mean    17.712560                 37.980193
std     11.392485                 13.606488
min      0.000000                  7.600000
25%      9.025000                 27.700000
50%     16.100000                 38.450000
75%     28.150000                 46.600000
max     43.800000                117.500000

       Distance to the nearest MRT station
count                           414.000000
mean                           1083.885689
std                            1262.109595
```

```
min                                  23.382840
25%                                 289.324800
50%                                 492.231300
75%                                1454.279000
max                                6488.021000
```

[13]:
```
# Compute correlation between age, distance, and price:
correlation = df[['House age', 'House price of unit area','Distance to the␣
 ↪nearest MRT station']].corr()
correlation
```

[13]:
```
                                    House age  House price of unit area  \
House age                            1.000000                 -0.210567
House price of unit area            -0.210567                  1.000000
Distance to the nearest MRT station  0.025622                 -0.673613

                                    Distance to the nearest MRT station
House age                                                      0.025622
House price of unit area                                      -0.673613
Distance to the nearest MRT station                            1.000000
```

[15]:
```
# Define X (Independent Variables) and y (Target Variable)
X = df[['House age', 'Distance to the nearest MRT station']]
y = df['House price of unit area']
```

[17]:
```
# Split data into training (70%) and testing (30%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=100)
```

[19]:
```
# Initialize and fit the Linear Regression model
LR = LinearRegression()
LR.fit(X_train, y_train)
```

[19]:
```
LinearRegression()
```

[21]:
```
# Assign variables for Predictions
LR_Predictions_Train = LR.predict(X_train)
LR_Predictions_Test = LR.predict(X_test)
```

[23]:
```
# Compute residuals and assign variables
residuals_train = y_train - LR_Predictions_Train
residuals_test = y_test - LR_Predictions_Test
```

[25]:
```
# Model Performance Metrics
# R^2 values for test and train
r2_train = r2_score(y_train, LR_Predictions_Train)
r2_test = r2_score(y_test, LR_Predictions_Test)
```

[27]:
```
n_train, k = X_train.shape
n_test = X_test.shape[0]
```

[29]:
```
# Obtain adjusted R^2 values for test and train
adj_r2_train = 1 - (1 - r2_train) * ((n_train - 1) / (n_train - k - 1))
adj_r2_test = 1 - (1 - r2_test) * ((n_test - 1) / (n_test - k - 1))
```

[31]:
```
# Obtain Mean Square Error for test and train
mse_train = mean_squared_error(y_train, LR_Predictions_Train)
mse_test = mean_squared_error(y_test, LR_Predictions_Test)
```

[33]:
```
# Obtain Root Mean Square Error for test and train
rmse_train = np.sqrt(mse_train)
rmse_test = np.sqrt(mse_test)
```

[35]:
```
# Add constant for statsmodels OLS summary
X_train_with_const = sm.add_constant(X_train)
ols_model = sm.OLS(y_train, X_train_with_const).fit()
summary_table = ols_model.summary()
```

0.2 Regression Model Comparison: Question 2 #10

0.2.1 Multiple linear regression model results

[37]:
```
# Print summary results all together
print("Scikit-learn Linear Regression Summary:")
print(f"Intercept: {LR.intercept_:.4f}")
print("Coefficients:")
print(pd.Series(LR.coef_, index=X_train.columns))

print("Training Set Performance:")
print(f"R-squared: {r2_train:.4f}")
print(f"Adjusted R-squared: {adj_r2_train:.4f}")
print(f"Mean Squared Error (MSE): {mse_train:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse_train:.4f}")

print("Test Set Performance:")
print(f"R-squared: {r2_test:.4f}")
print(f"Adjusted R-squared: {adj_r2_test:.4f}")
print(f"Mean Squared Error (MSE): {mse_test:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse_test:.4f}")

# Display full statsmodels-style summary
print("Statsmodels OLS Summary (Training Data):")
print()
print(summary_table)
```

```
Scikit-learn Linear Regression Summary:
Intercept: 50.2065
Coefficients:
House age                               -0.244615
Distance to the nearest MRT station     -0.007079
dtype: float64
Training Set Performance:
R-squared: 0.4750
Adjusted R-squared: 0.4713
Mean Squared Error (MSE): 106.8896
Root Mean Squared Error (RMSE): 10.3387
Test Set Performance:
R-squared: 0.5401
Adjusted R-squared: 0.5326
Mean Squared Error (MSE): 64.4566
Root Mean Squared Error (RMSE): 8.0285
Statsmodels OLS Summary (Training Data):

                              OLS Regression Results
================================================================================
====
Dep. Variable:     House price of unit area   R-squared:
0.475
Model:                                  OLS   Adj. R-squared:
0.471
Method:                       Least Squares   F-statistic:
129.4
Date:                      Sun, 02 Mar 2025   Prob (F-statistic):
9.71e-41
Time:                              08:21:08   Log-Likelihood:
-1085.1
No. Observations:                       289   AIC:
2176.
Df Residuals:                           286   BIC:
2187.
Df Model:                                 2
Covariance Type:                  nonrobust
================================================================================
=======================
                                          coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
-----------------------
const                                  50.2065      1.246     40.295      0.000
47.754      52.659
House age                              -0.2446      0.053     -4.626      0.000
-0.349      -0.141
Distance to the nearest MRT station    -0.0071      0.000    -15.251      0.000
```

```
-0.008      -0.006
==============================================================================
Omnibus:                      126.749   Durbin-Watson:                   1.948
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1026.193
Skew:                           1.573   Prob(JB):                    1.46e-223
Kurtosis:                      11.679   Cond. No.                     3.56e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.56e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[41]:

```python
# Import dependencies for Variance Inflation Factor to check if not over
 ↪correlated. Results show that there is
# no multicollinearity since the results are near 1 for the applicable features
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Add constant for VIF calculation
X_with_const = sm.add_constant(X)

# Calculate VIF for each predictor
vif_data = pd.DataFrame()
vif_data["Feature"] = X_with_const.columns
vif_data["VIF"] = [variance_inflation_factor(X_with_const.values, i) for i in
 ↪range(X_with_const.shape[1])]

# Display VIF values
print(vif_data)
```

```
                                Feature       VIF
0                                 const  4.095876
1                             House age  1.000657
2  Distance to the nearest MRT station  1.000657
```

[45]:

```python
from scipy import stats

# Given values (from statsmodels output)
beta_1 = -0.244615  # House Age
std_err_1 = 0.053
beta_2 = -0.007079  # Distance to MRT
std_err_2 = 0.00047
df = len(X_train) - 2  # Degrees of freedom (n - k - 1, k=2 for two predictors)

# Compute t-statistic and p-value for both predictors
t_statistic_1 = beta_1 / std_err_1
```

```python
p_value_1 = 2 * (1 - stats.t.cdf(abs(t_statistic_1), df))

t_statistic_2 = beta_2 / std_err_2
p_value_2 = 2 * (1 - stats.t.cdf(abs(t_statistic_2), df))

print("Multiple Regression - House Age & Distance to MRT")
print(f"House Age: T-statistic = {t_statistic_1:.3f}, P-value = {p_value_1:.
 ↪6f}")
print(f"Distance to MRT: T-statistic = {t_statistic_2:.3f}, P-value =␣
 ↪{p_value_2:.6f}")
```

```
Multiple Regression - House Age & Distance to MRT
House Age: T-statistic = -4.615, P-value = 0.000006
Distance to MRT: T-statistic = -15.062, P-value = 0.000000
```

[ ]:

```python

```

hw_3_question3

March 3, 2025

[3]:
```python
import pandas as pd
import numpy as np
import scipy.stats as st
```

[5]:
```python
# Created an array of the values
fastfood = np.array([7.42,6.29,5.83,6.50,8.34,9.51,7.10,6.80,5.90,4.89,6.50,5.
↪52,7.90,8.30,9.60])
fastfood
```

[5]:
```
array([7.42, 6.29, 5.83, 6.5 , 8.34, 9.51, 7.1 , 6.8 , 5.9 , 4.89, 6.5 ,
       5.52, 7.9 , 8.3 , 9.6 ])
```

[7]:
```python
# Determine the mean
mean = np.mean(fastfood)
mean
```

[7]:
```
7.093333333333333
```

[9]:
```python
# Capture the standard deviation
std_dev = np.std(fastfood, ddof=1)
std_dev
```

[9]:
```
1.4060312263686783
```

[11]:
```python
# Calculate the standard error
standard_error = std_dev / np.sqrt(len(fastfood))
standard_error
```

[11]:
```
0.3630357015982321
```

[13]:
```python
# Assign the required confidence level
confidence_level = .95
```

[15]:
```python
# calculate degrees of freedom to reduce bias
degrees_of_freedom = len(fastfood) - 1
```

[17]:
```
# detemrine the critical value
critical_value = st.t.ppf((1 + confidence_level) / 2, degrees_of_freedom)
critical_value
```

[17]: 2.1447866879169273

[19]:
```
margin_of_error = critical_value * standard_error
margin_of_error
```

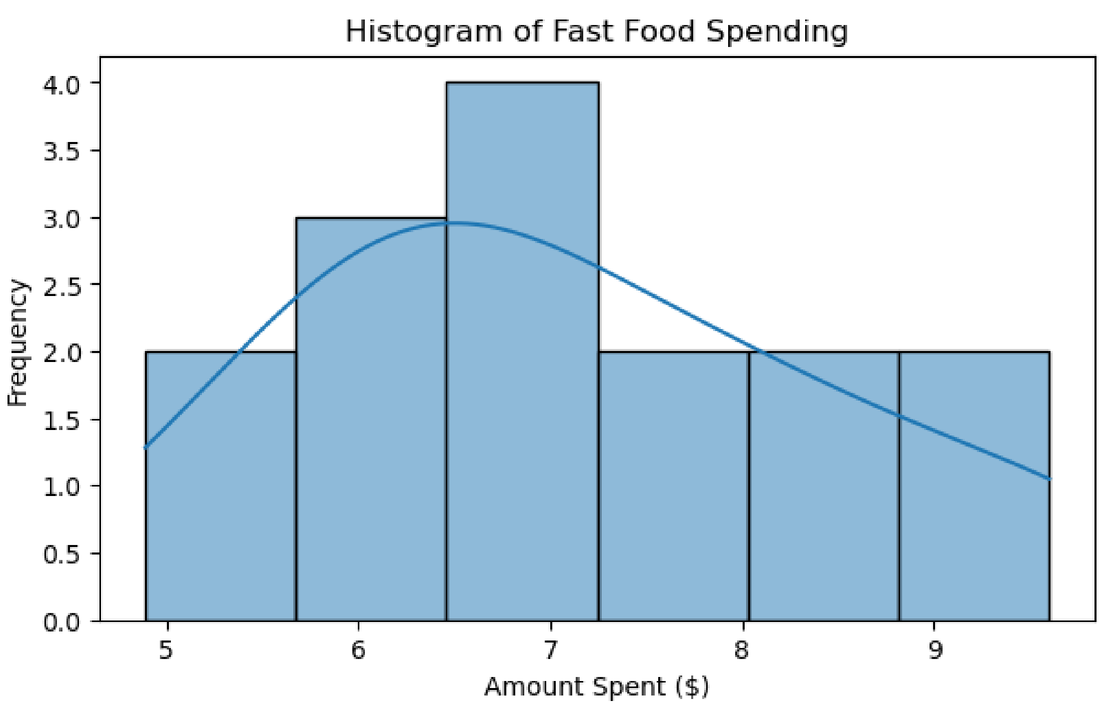
[19]: 0.7786341400264702

[21]:
```
# determine the range of the confidence interval
confidence_interval = (mean - margin_of_error, mean + margin_of_error)
confidence_interval
```

[21]: (6.314699193306863, 7.871967473359803)

[23]:
```
import matplotlib.pyplot as plt
```

[27]:
```
import seaborn as sns
plt.figure(figsize=(7,4))
sns.histplot(fastfood, bins=6, kde=True)
plt.xlabel("Amount Spent ($)")
plt.ylabel("Frequency")
plt.title("Histogram of Fast Food Spending")
plt.show()
```



[31]:
```python
# using p value (from ChatGPT)

import scipy.stats as stats
import numpy as np


# Hypotheses:
# H0: mean = 6.50 (The mean spending is $6.50)
# H1: mean != 6.50 (The mean spending is not $6.50)

mu_0 = 6.50  # Population mean to test against
t_stat, p_value = stats.ttest_1samp(fastfood, mu_0)

# Print results
print(f"T-statistic: {t_stat:.3f}")
print(f"P-value: {p_value:.4f}")

# Decision at alpha = 0.05
alpha = 0.05
if p_value < alpha:
    print("Reject H0: There is significant evidence that mean spending is␣
↪different from $6.50.")
else:
    print("Fail to reject H0: Not enough evidence to say spending is different␣
↪from $6.50.")
```

```
T-statistic: 1.634
P-value: 0.1245
Fail to reject H0: Not enough evidence to say spending is different from $6.50.
```

[35]:
```python
# using test statistics (from ChatGPT)
mu_0 = 6.50

# Sample statistics
n = len(fastfood)
mean_x = np.mean(fastfood)
std_x = np.std(fastfood, ddof=1)  # Sample standard deviation

# Compute t-test statistic
t_stat = (mean_x - mu_0) / (std_x / np.sqrt(n))

# Find critical t-value (two-tailed test at alpha = 0.05)
alpha = 0.05
t_critical = stats.t.ppf(1 - alpha/2, df=n-1)
```

```python
# Print results
print(f"Sample Mean: {mean_x:.3f}")
print(f"Sample Standard Deviation: {std_x:.3f}")
print(f"T-statistic: {t_stat:.3f}")
print(f"Critical T-value (two-tailed at alpha=0.05): +-{t_critical:.3f}")
```

```
Sample Mean: 7.093
Sample Standard Deviation: 1.406
T-statistic: 1.634
Critical T-value (two-tailed at alpha=0.05): +-2.145
```

[ ]:

```python

```