HW2: Exploratory Data Analysis on an insurance spending dataset

Learning the data using visualization and a simple linear regression

1. First, run the example and understand the ML process and be familir with Python package functions for simple linear regression.

2. Apply exploratory data analysis and simple regression on the insurance data: 'insurance.csv'

Write your name

- Priyadarshini Shanmugasundaram Murugan

Simple Linear Regression example

Follow the code and observe the results

```
# you need   Python ≥3.5
import sys
assert sys.version_info >= (3, 5)

# Scikit-Learn ≥0.20 is required
import sklearn
assert sklearn.__version__ >= "0.20"
```

==========

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
import warnings
warnings.filterwarnings("ignore")
##############################################
df = pd.read_csv('insurance.csv')
df.head()
```

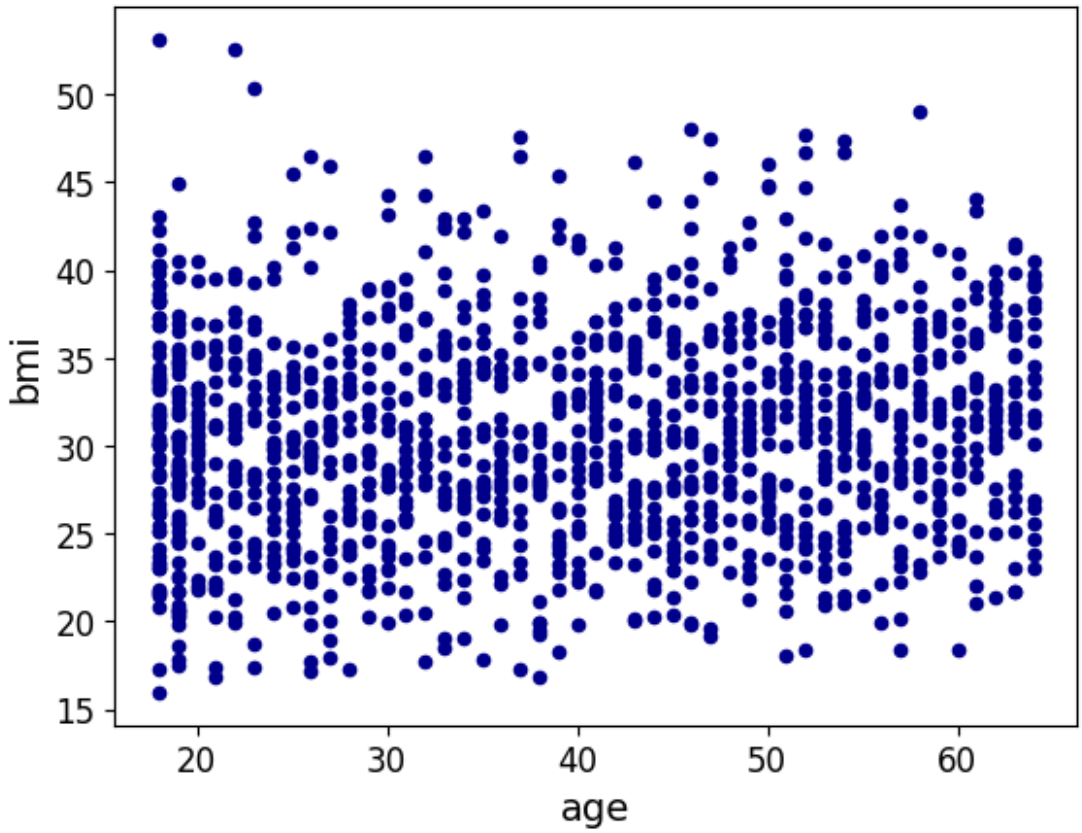
	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	2320.00
1	18	male	33.770	1	no	southeast	2506.10
2	28	male	33.000	3	no	southeast	2512.00

```
3   33    male  22.705         0     no  northwest  2124.15
4   32    male  28.880         0     no  northwest  2407.40
```

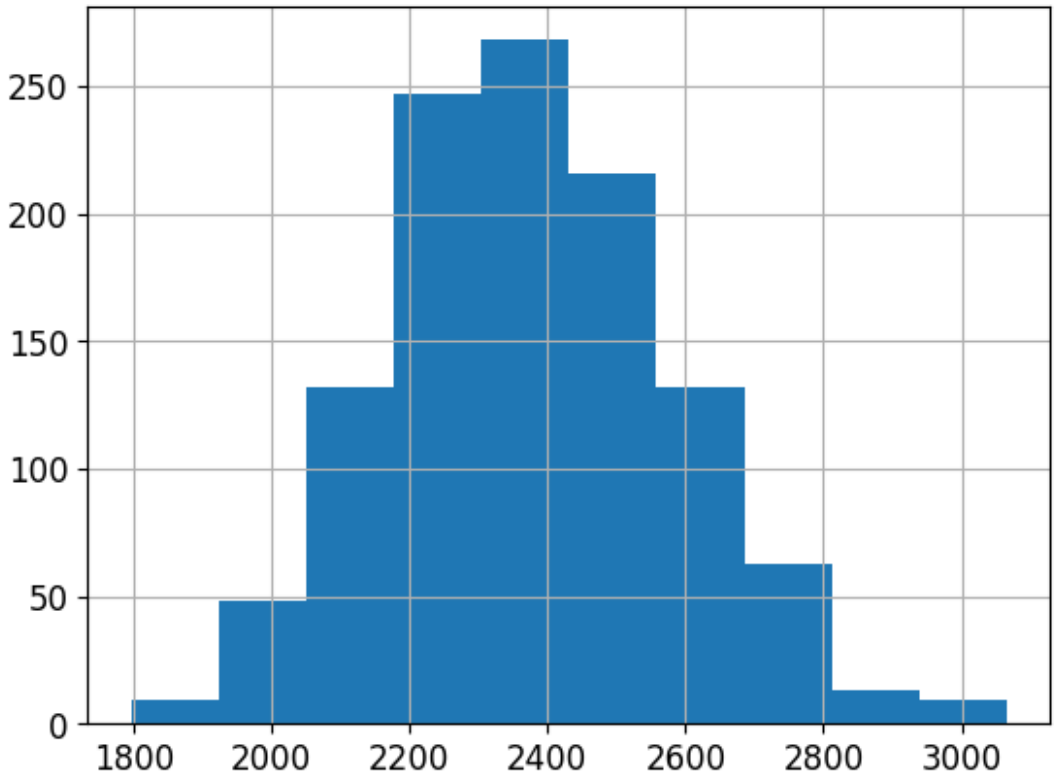
```
# To plot  figures directly within Jupyter
%matplotlib inline
import matplotlib as mpl
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)
```

Plot scatterplot and the regression function

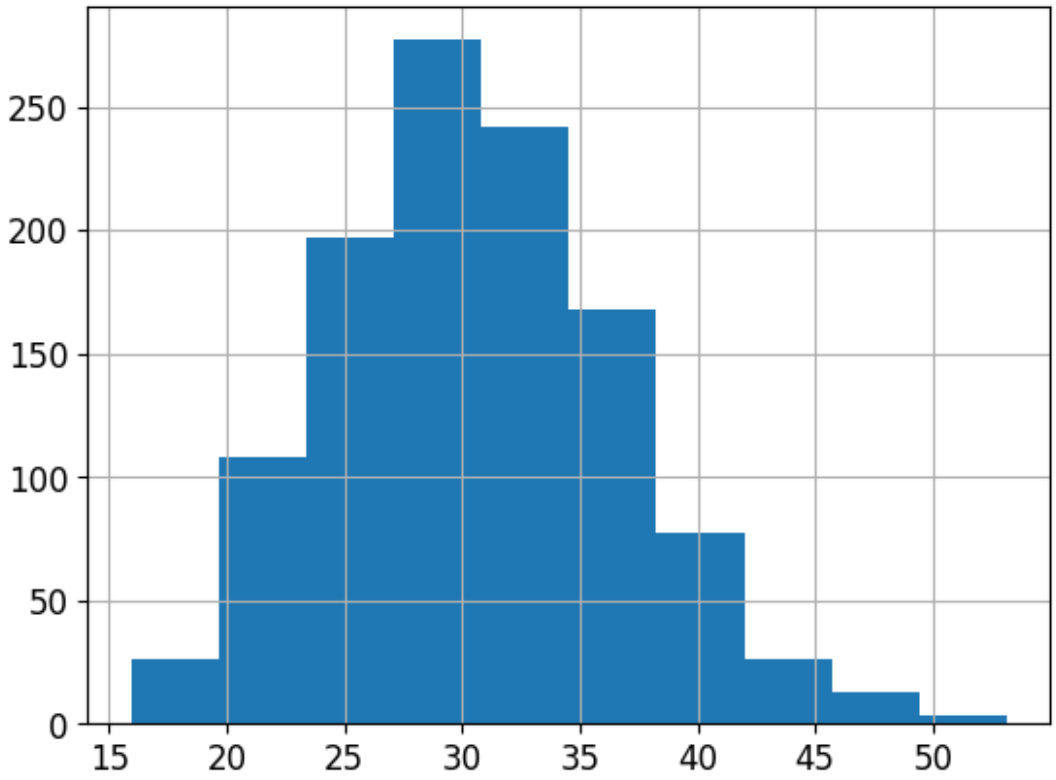
```
# Code example of load the data and prepared the data
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn.linear_model
ax1 = df.plot.scatter(x='age',
                      y='bmi',
                      c='DarkBlue')
plt.show()
```




```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
# split the data into train and test parts
train_df, test_df = train_test_split(df,shuffle = True, test_size =
0.15, random_state=17)
# show descriptive charachteristics of the dataset such as min, max
median

train_df['charges'].hist()
plt.show()
train_df['bmi'].hist()
```




```
<Axes: >
```

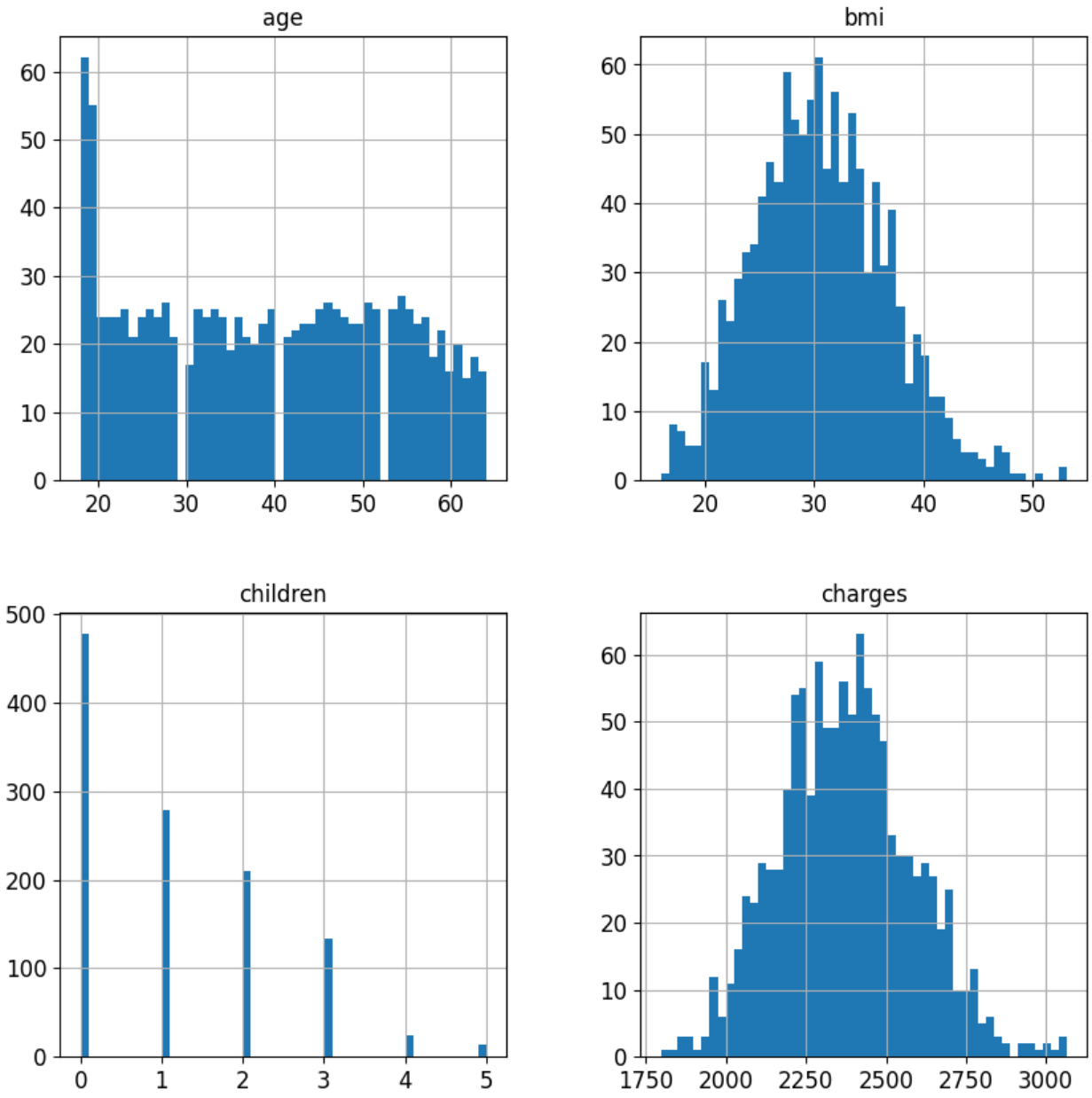



```
numerical_features=list(train_df.columns)
print(numerical_features)
short_list_features = ['age', 'bmi', 'children', 'charges']
print(short_list_features)
```

```
['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges']
['age', 'bmi', 'children', 'charges']
```

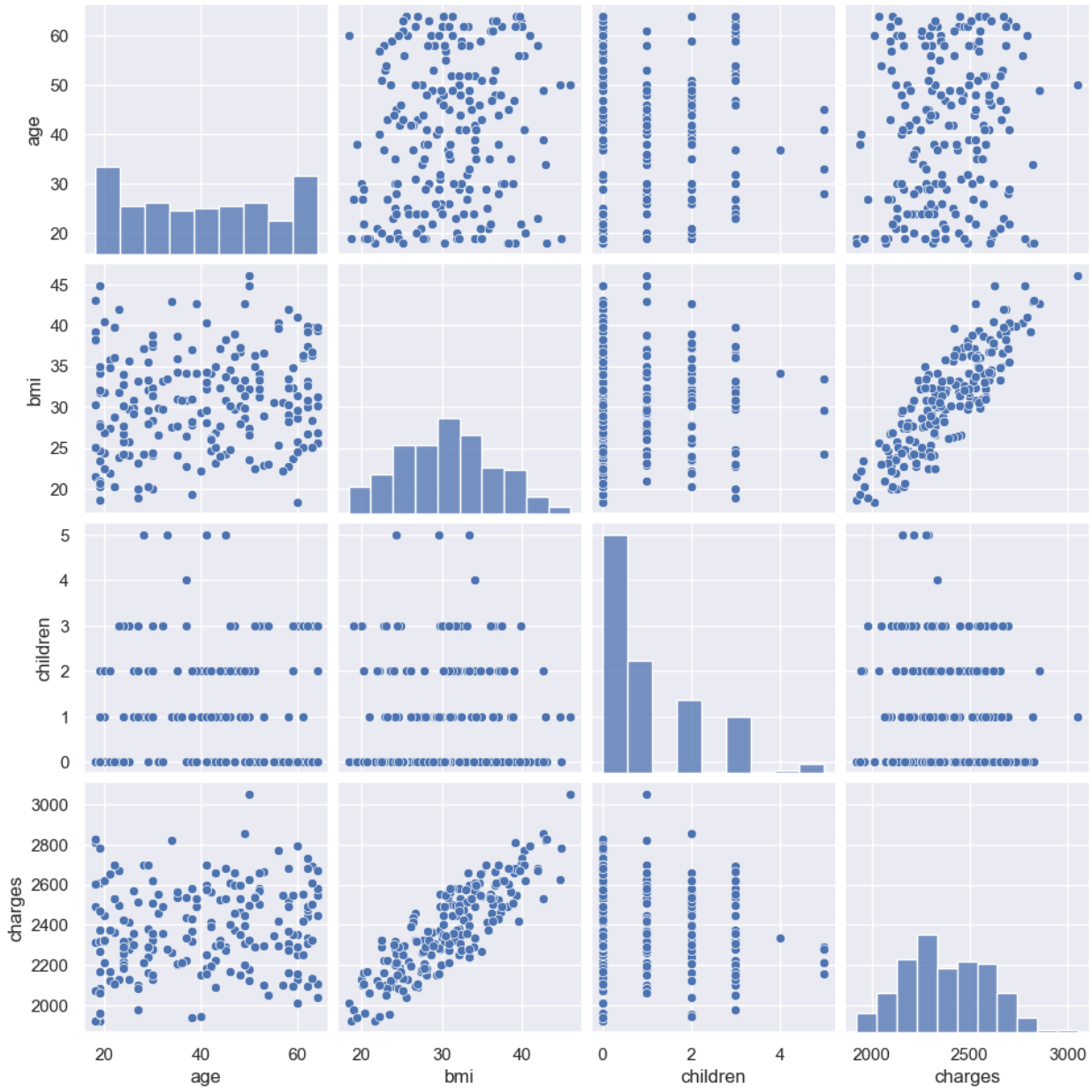
```
train_df[short_list_features ].hist(bins=50, figsize=(10, 10))
```

```
array([[<Axes: title={'center': 'age'}>, <Axes: title={'center':
'bmi'}>],
       [<Axes: title={'center': 'children'}>,
        <Axes: title={'center': 'charges'}>]], dtype=object)
```




```
sns.set()
sns.pairplot(test_df[short_list_features])
```

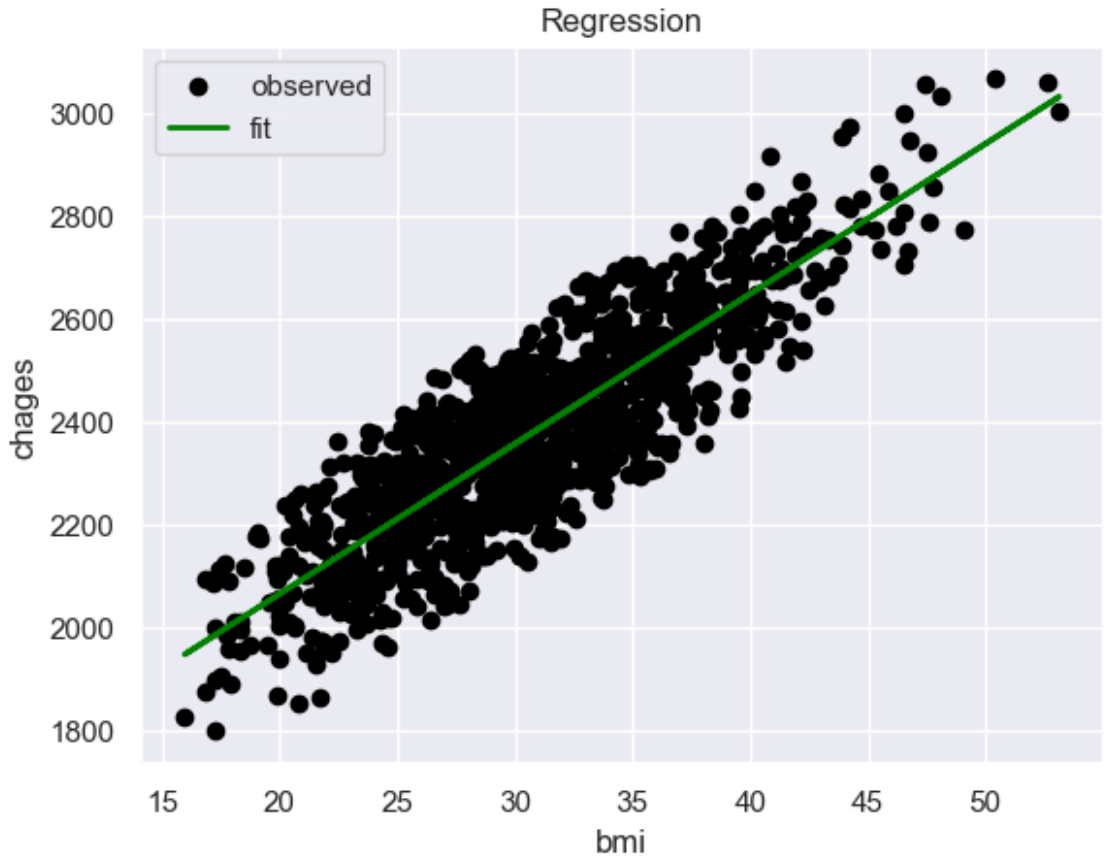
```
<seaborn.axisgrid.PairGrid at 0x1f12f9ad280>
```




```
# Scikit-Learn ≥0.20 is required
import sklearn
from sklearn import linear_model
# Select a linear model
import numpy as np
from sklearn.linear_model import LinearRegression

X= train_df['bmi']
y = train_df['charges']
X2= np.array(X).reshape(-1,1) # reshape(-1,1) makes the array
horizontal, y is vertical
```

```
# Train the model (we use all data for training -->
model = sklearn.linear_model.LinearRegression()
model.fit(X2, y)
r_sq = model.score(X2, y)
print(r_sq)
plt.scatter(X2, y, color='black', label='observed')

plt.plot(X2, model.predict(X2), label='fit', color='Green',
linewidth=2)

plt.xlabel('bmi')
plt.ylabel('chages')
plt.title('Regression')
plt.legend(loc='best')

plt.show()
```

```
0.7437862976848658
```




```
# Make a prediction
X3= test_df['bmi']

X3= np.array(X3).reshape(-1,1)
```

```
# what is the charges for element 44 of the test bmi list.
predicted_charge = model.predict([X3[44]])
print('bmi= {:.2f}   predicted charge = {:.2f}'.format(X3[44][0],
predicted_charge[0]))
```

```
bmi= 37.29   predicted charge = 2568.86
```

Assignment

Part 1: Explore insight of the data using other visualization tools

'Real estate.csv'

Tasks

1. Select attributes (columns) you are interested .
2. Conduct exploratory data analysis on the selected data and visualize the data. This may include the following but not limited: (Data should be cleaned. For example, zeros in inapropriate locations should be replaced by mean or median values.)

(a) The mean, median and standard deviation

(b) Draw boxplots

(c) Draw histograms

(c) Draw scatter plots

(d) simple linear regression function with a scatter plot

Real Estate Prediction

```
# load the data and prepared the data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv('Real estate.csv')
df.head()
```

```
   No  X1 transaction date  X2 house age  X3 bus station  X4 number of
stores  \
0   1             2012.917          32.0        84.87882
10
1   2             2012.917          19.5       306.59470
9
```

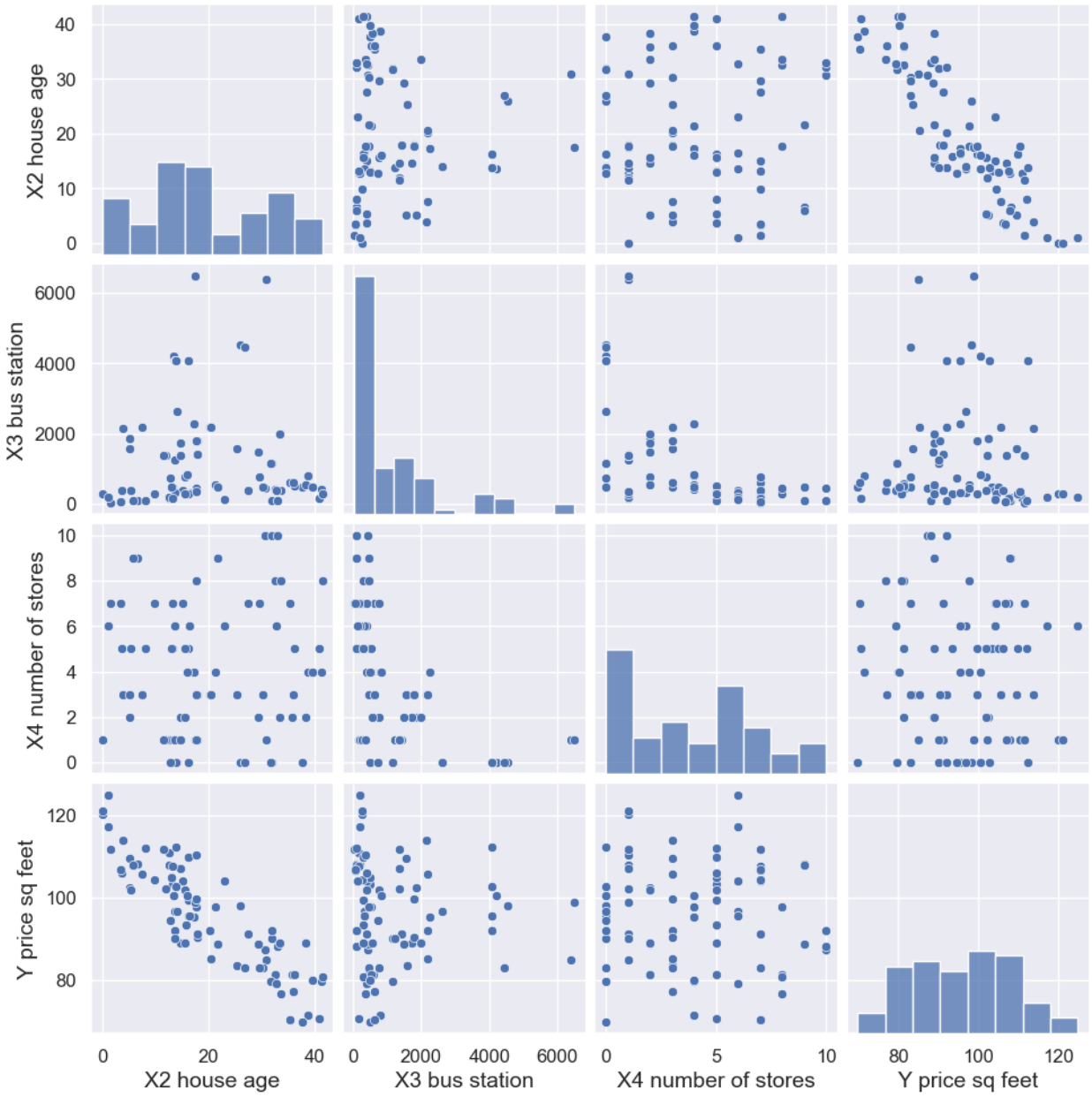
```
2   3             2013.583          13.3       561.98450
5
3   4             2013.500          13.3       561.98450
5
4   5             2012.833           5.0       390.56840
5

   X5 latitude  X6 longitude  Y price sq feet
0     24.98298     121.54024             92.1
1     24.98034     121.53951             82.4
2     24.98746     121.54391             93.6
3     24.98746     121.54391             97.3
4     24.97937     121.54245             99.9
```

```
# To plot  figures directly within Jupyter
%matplotlib inline
import matplotlib as mpl
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

# split the data into train and test parts
train_df, test_df = train_test_split(df,shuffle = True, test_size =
0.2, random_state=42)
# show descriptive charachteristics of the dataset such as min, max
median
description = df.describe()
print("Descriptve Characteristics :")
print(description)
```

```
Descriptve Characteristics :
               No  X1 transaction date  X2 house age  X3 bus
station  \
count  414.000000           414.000000    414.000000      414.000000

mean   207.500000          2013.148971     17.712560     1083.885689

std    119.655756             0.281967     11.392485     1262.109595

min      1.000000          2012.667000      0.000000       23.382840

25%    104.250000          2012.917000      9.025000      289.324800

50%    207.500000          2013.167000     16.100000      492.231300

75%    310.750000          2013.417000     28.150000     1454.279000

max    414.000000          2013.583000     43.800000     6488.021000


       X4 number of stores  X5 latitude  X6 longitude  Y price sq feet
```

```
count            414.000000    414.000000    414.000000        414.000000
mean               4.094203     24.908709    121.239786         97.754106
std                2.945562      1.227223      5.973058         12.850456
min                0.000000      0.000000      0.000000         65.300000
25%                1.000000     24.962990    121.527600         89.025000
50%                4.000000     24.971100    121.538630         99.350000
75%                6.000000     24.977455    121.543305        106.675000
max               10.000000     25.014590    121.566270        127.000000
```

```
#pair plotting
import seaborn as sns

# Select the columns you want to include in the pair plot
columns_to_plot = ['X2 house age', 'X3 bus station', 'X4 number of
stores', 'Y price sq feet']

# Create a pair plot
sns.pairplot(test_df[columns_to_plot])

# Show the plot
plt.show()
```



Pairplots offer a convenient method for visualizing relationships between variables within a dataset. By examining scatterplots, analysts can swiftly identify both linear and non-linear relationships between pairs of variables. Pairplots aid in pinpointing correlations between variables, with clear patterns observable in the scatterplots, such as points forming straight lines between 'X2 house age' and 'Y price sq feet'. Consequently, the columns 'X2 house age' and 'Y price sq feet' are selected for conducting exploratory data analysis and visualizing the data.

```
#Before Cleaning Data

# Calculate and print Mean, Median, and Standard Deviation for House Age
print("Before Cleaning Data")
```

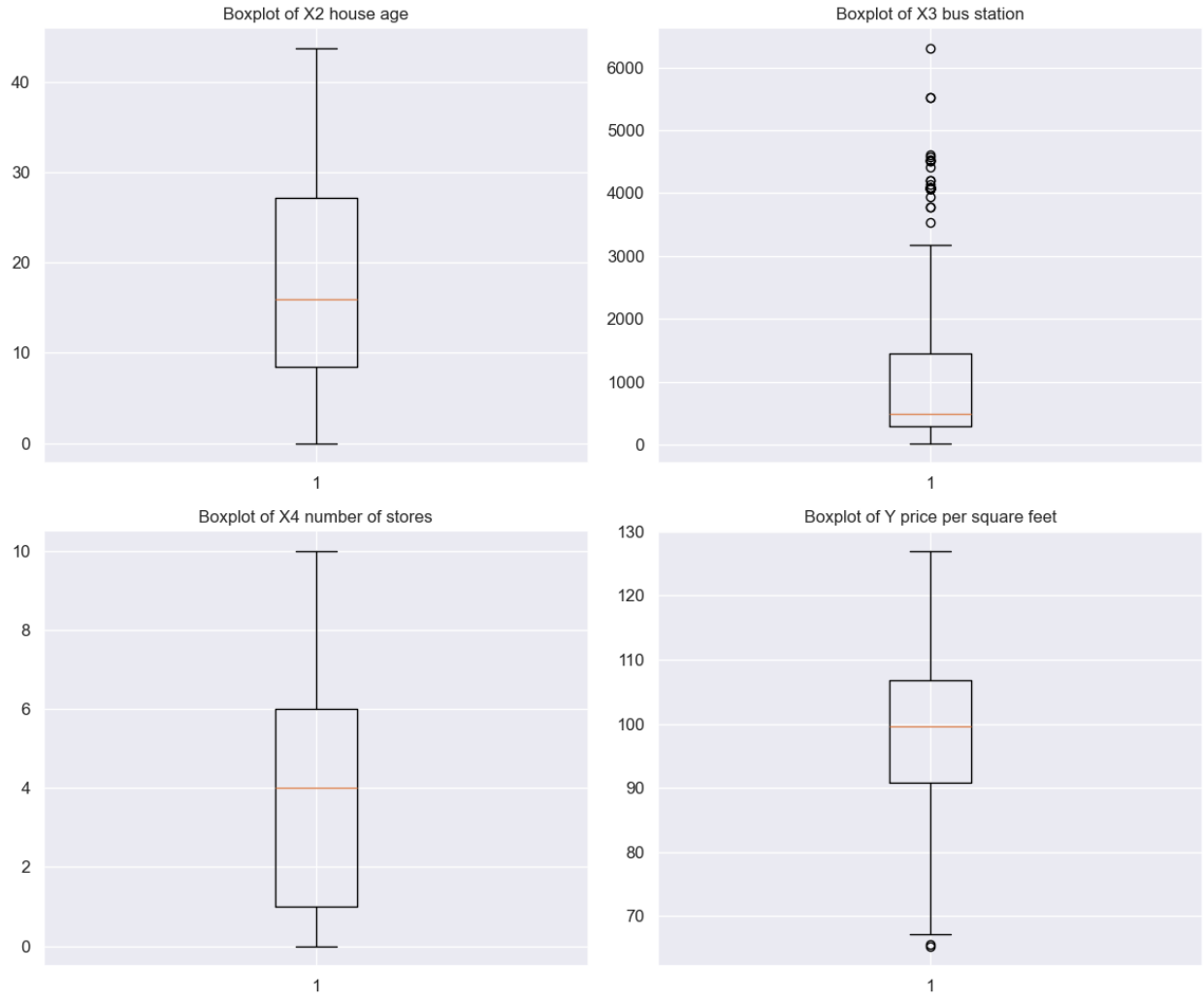
```python
print("Mean House Age:", df['X2 house age'].mean())
print("Median House Age:", df['X2 house age'].median())
print("Standard Deviation House Age:", df['X2 house age'].std())

# Calculate and print Mean, Median, and Standard Deviation for House
Price
print("\nMean House Price:", df['Y price sq feet'].mean())
print("Median House Price:", df['Y price sq feet'].median())
print("Standard Deviation House Price:", df['Y price sq feet'].std())

# Selecting specific columns
selected_data = df[['X2 house age', 'Y price sq feet']]

# Replace zeros in column 'X2 house age' with the mean value
selected_data['X2 house age'] = selected_data['X2 house
age'].replace(0, selected_data['X2 house age'].mean())

# Replace zeros in column 'Y price sq feet' with the median value
selected_data['Y price sq feet'] = selected_data['Y price sq
feet'].replace(0, selected_data['Y price sq feet'].median())

print("\nAfter cleaning data")
# Mean
mean = selected_data.mean().to_dict()
print("\nMean:\n", mean)

# Median
median = selected_data.median().to_dict()
print("\nMedian:\n", median)

# Standard deviation
std_dev = selected_data.std().to_dict()
print("\nStandard Deviation:\n", std_dev)
```

```
Before Cleaning Data
Mean House Age: 17.71256038647343
Median House Age: 16.1
Standard Deviation House Age: 11.39248453324253

Mean House Price: 97.75410628019323
Median House Price: 99.35
Standard Deviation House Price: 12.850455566991938

After cleaning data

Mean:
 {'X2 house age': 18.439887745338282, 'Y price sq feet':
97.75410628019323}

Median:
 {'X2 house age': 16.5, 'Y price sq feet': 99.35}
```

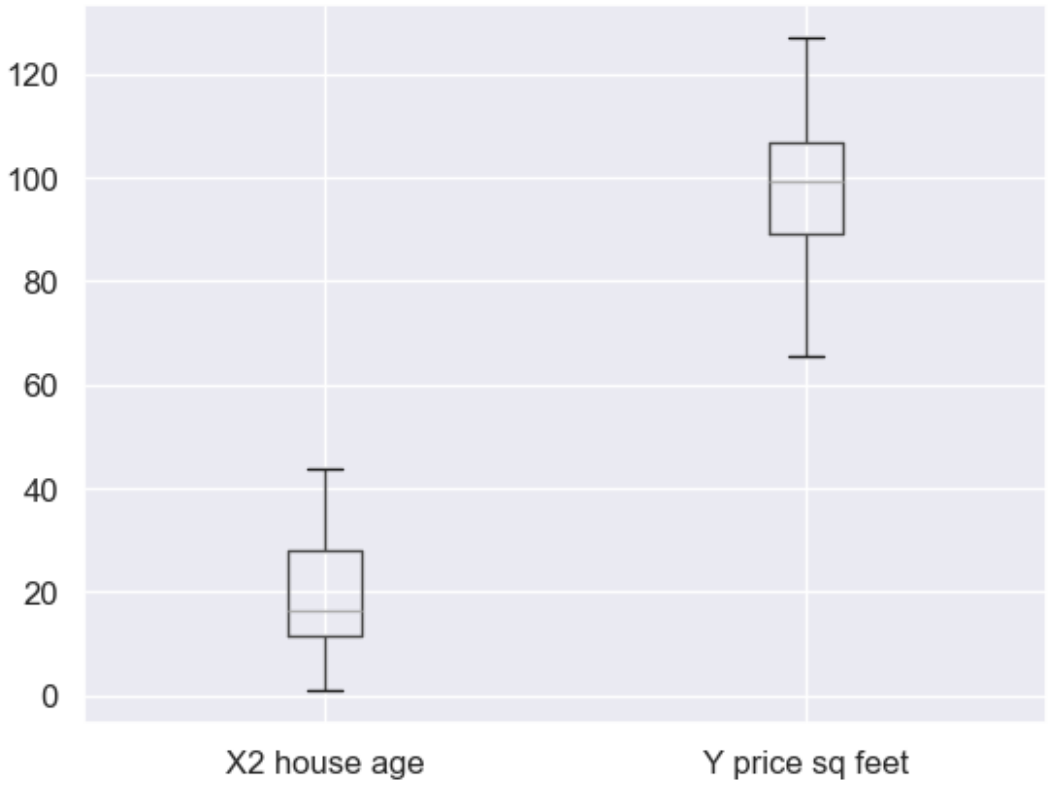
```
Standard Deviation:
 {'X2 house age': 10.786305892760181, 'Y price sq feet':
12.850455566991938}
```

Data cleaning plays a pivotal role in preparing datasets for analysis and modeling. By substituting zeros with appropriate summary statistics like mean or median, we ensure data is more representative and conducive to further analysis. Moreover, computing descriptive statistics aids in comprehending data distribution and properties, crucial for making informed decisions during analysis.replacing zeros with appropriate summary statistics.

```
# Create a figure and axis object
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# Plot box plot for X2 house age
axes[0, 0].boxplot(train_df['X2 house age'])
axes[0, 0].set_title('Boxplot of X2 house age')

# Plot box plot for X3 bus station
axes[0, 1].boxplot(train_df['X3 bus station'])
axes[0, 1].set_title('Boxplot of X3 bus station')

# Plot box plot for X4 number of stores
axes[1, 0].boxplot(train_df['X4 number of stores'])
axes[1, 0].set_title('Boxplot of X4 number of stores')

# Plot box plot for Y price per square feet
axes[1, 1].boxplot(train_df['Y price sq feet'])
axes[1, 1].set_title('Boxplot of Y price per square feet')

# Adjust layout
plt.tight_layout()
plt.show()
```



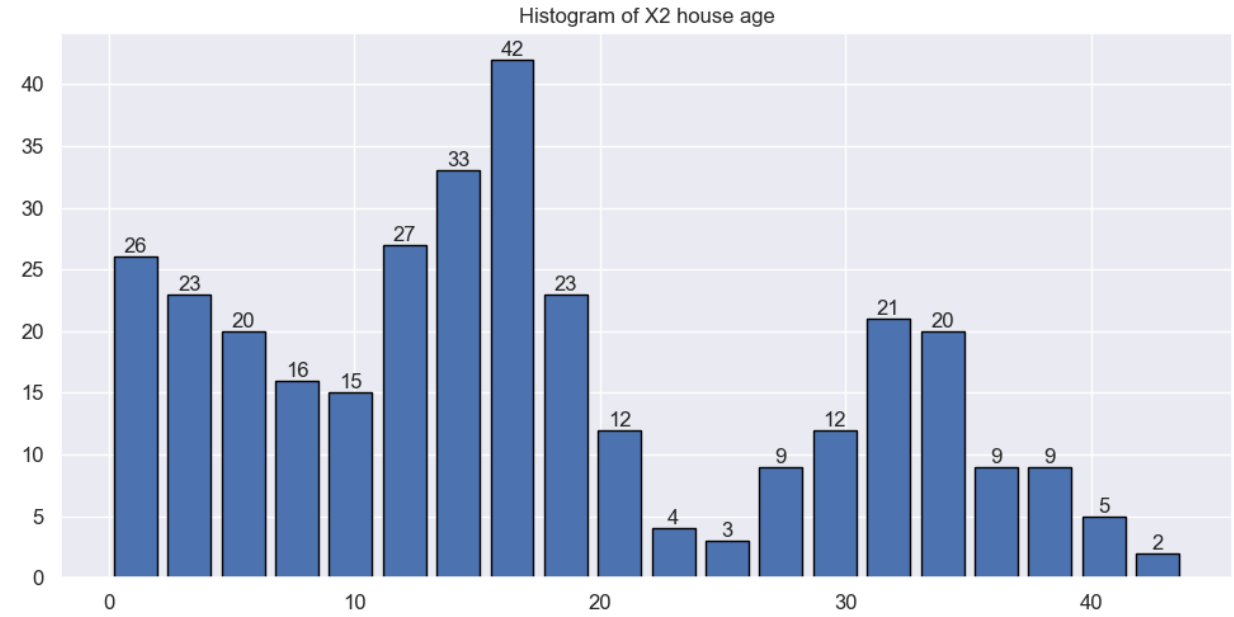

```
#Boxplot for selected data

selected_data.boxplot()
plt.show()
```

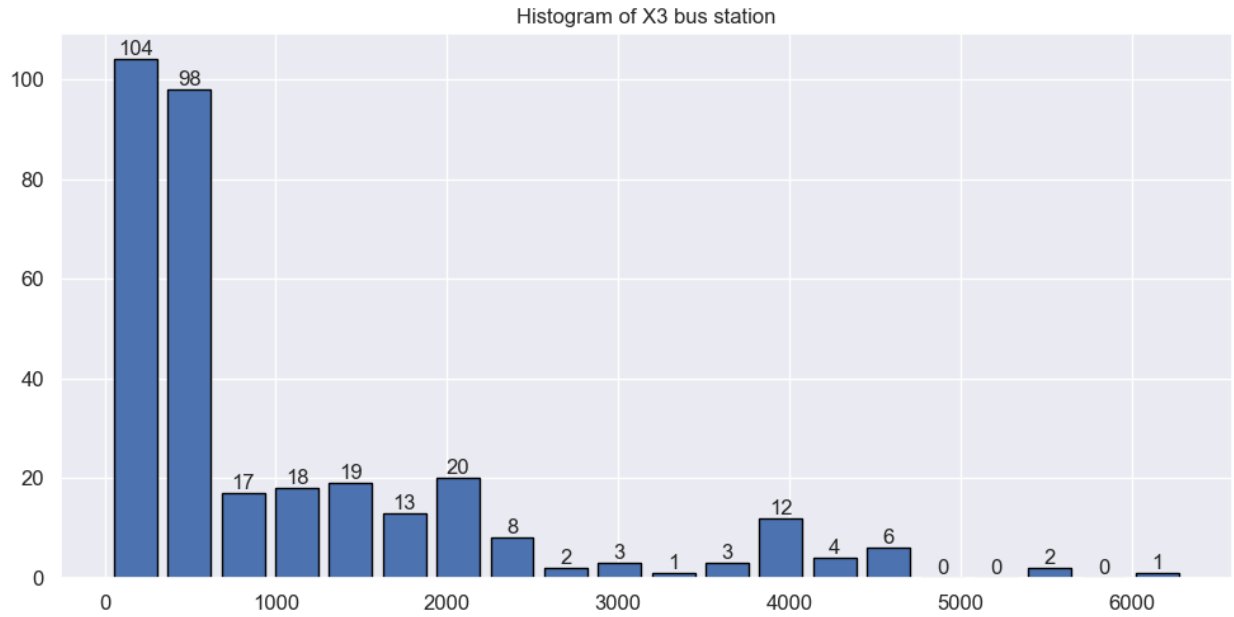


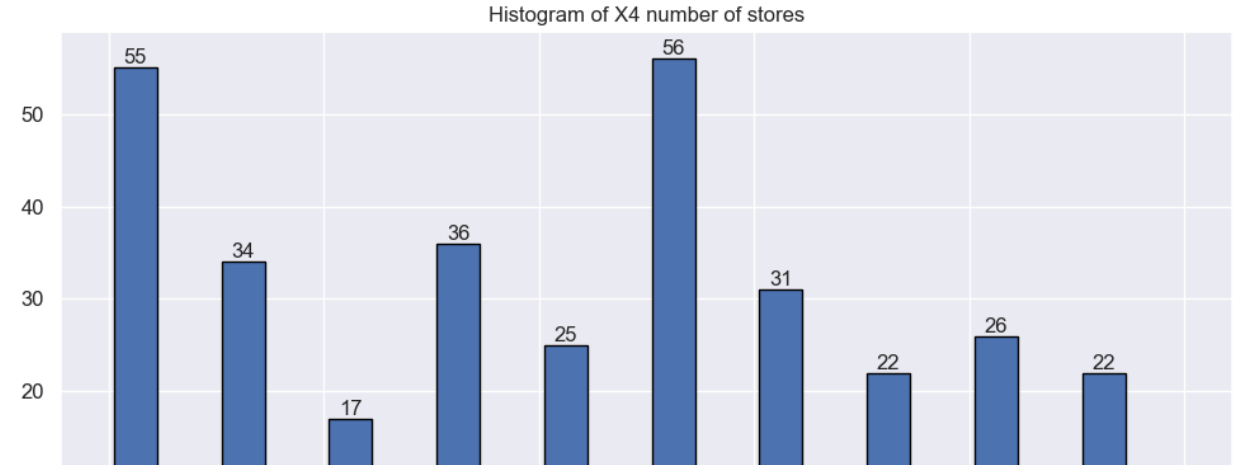
Boxplots offer a concise summary of data distribution, central tendency, and variability, aiding in identifying patterns, anomalies, and potential issues. Subplots allow for clear comparisons between variables, facilitating exploratory data analysis. The plt.tight_layout() function ensures proper spacing for readability. Overall, boxplots provide an intuitive visualization of data across different attributes, supporting decision-making processes in data analysis.

```
# Histogram
fig, axes = plt.subplots(4, 1, figsize=(10, 20))

# Plot histogram for X2 house age
train_df['X2 house age'].hist(bins=20, ax=axes[0], edgecolor='black',
rwidth=0.8)
axes[0].set_title('Histogram of X2 house age')
for i in axes[0].patches:
    axes[0].text(i.get_x() + i.get_width()/2, i.get_height(),
str(int(i.get_height())), ha='center', va='bottom')

# Plot histogram for X3 bus station
train_df['X3 bus station'].hist(bins=20, ax=axes[1],
edgecolor='black', rwidth=0.8)
axes[1].set_title('Histogram of X3 bus station')
for i in axes[1].patches:
    axes[1].text(i.get_x() + i.get_width()/2, i.get_height(),
str(int(i.get_height())), ha='center', va='bottom')
```

```
# Plot histogram for X4 number of stores
train_df['X4 number of stores'].hist(bins=20, ax=axes[2],
edgecolor='black', rwidth=0.8)
axes[2].set_title('Histogram of X4 number of stores')
for i in axes[2].patches:
    axes[2].text(i.get_x() + i.get_width()/2, i.get_height(),
str(int(i.get_height())), ha='center', va='bottom')

# Plot histogram for Y price per square feet
train_df['Y price sq feet'].hist(bins=20, ax=axes[3],
edgecolor='black', rwidth=0.8)
axes[3].set_title('Histogram of Y price per square feet')
for i in axes[3].patches:
    axes[3].text(i.get_x() + i.get_width()/2, i.get_height(),
str(int(i.get_height())), ha='center', va='bottom')

# Adjust layout
plt.tight_layout()
plt.show()
```


Histogram of X2 house age
0
5
10
15
20
25
30
35
40
0
10
20
30
40
26
23
20
16
15
27
33
42
23
12
4
3
9
12
21
20
9
9
5
2

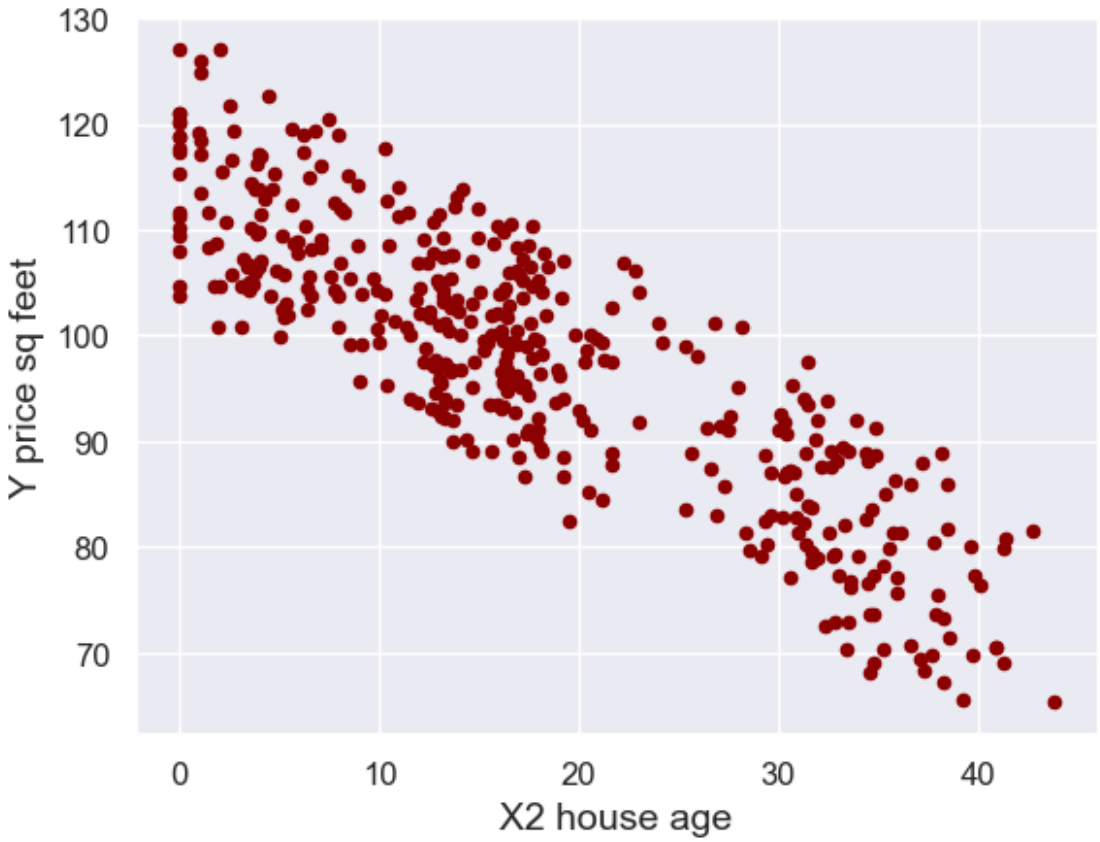

Histogram of X3 bus station
0
20
40
60
80
100
0
1000
2000
3000
4000
5000
6000
104
98
17
18
19
13
20
8
2
3
1
3
12
4
6
0
0
2
0
1


Histogram of X4 number of stores
20
30
40
50
55
34
17
36
25
56
31
22
26
22

```
# Selected data histogram
selected_data.hist()
plt.show()
```

Histograms offer a visual representation of the distribution of data within individual attributes. They provide insights into the central tendency, spread, and shape of the data's distribution, aiding in identifying patterns, outliers, and assessing skewness or symmetry. Analyzing histograms enables a deeper understanding of the dataset's underlying characteristics, facilitating informed decisions in subsequent data analysis processes.

```
# d) Scatter Points

ax1 = df.plot.scatter(x='X2 house age',
                      y='Y price sq feet',
                      c='DarkRed')
plt.show()
```



The essence of using scatter plots to explore the relationship between two continuous variables. By examining the pattern of data points on the plot, we can indeed assess various aspects of the relationship, including its strength, direction, and form. Additionally, scatter plots provide insights into the spread of data along both axes, helping us understand the variability in the dataset. By visualizing 'X2 house age' against 'Y price sq feet' with data points colored in dark red effectively portrays the relationship between house age and price per square feet.

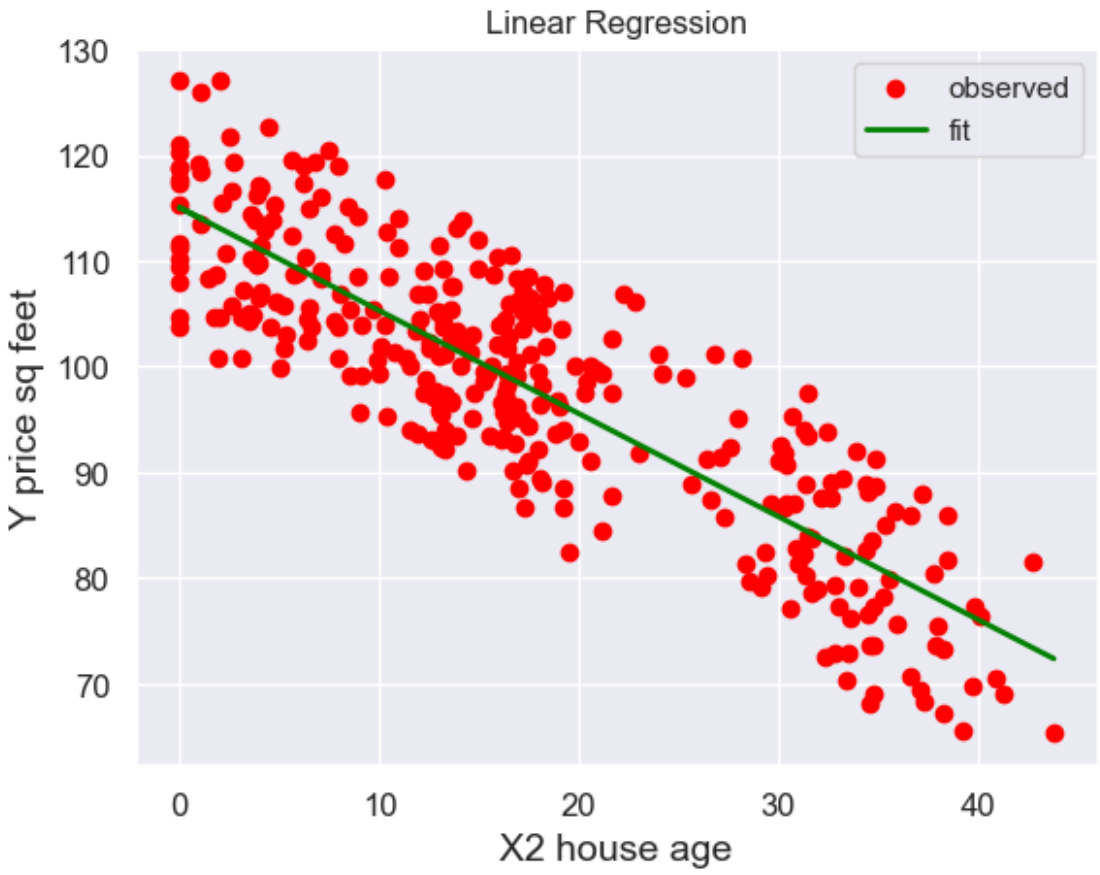
```
# Selecting the columns

X_columns = ['X2 house age', 'X3 bus station', 'X4 number of stores']
Y_column = 'Y price sq feet'

# Create a dictionary to store R-squared values
r_squared_values = {}

# Iterate over each pair of X and Y
for X_col in X_columns:
    X = train_df[X_col].values.reshape(-1, 1)
    Y = train_df[Y_column].values

    # Create and train the model
    model = LinearRegression()
    model.fit(X, Y)
```

```
    # Calculate R-squared
    r_squared = model.score(X, Y)

    # Store R-squared value in the dictionary
    r_squared_values[X_col] = r_squared

# Find the column with the highest R-squared value
max_col = max(r_squared_values, key=r_squared_values.get)
max_r_squared = r_squared_values[max_col]

# Print R-squared values
for X_col, r_squared in r_squared_values.items():
    if r_squared == max_r_squared:
        print(f"R-squared between {X_col} and {Y_column}: \
033[91m{r_squared}\033[0m")
    else:
        print(f"R-squared between {X_col} and {Y_column}:
{r_squared}")
```

```
R-squared between X2 house age and Y price sq feet: 0.7349929890443487
R-squared between X3 bus station and Y price sq feet:
0.0008745246625989633
R-squared between X4 number of stores and Y price sq feet:
0.004092106836189102
```

```
# e)simple linear regression function with a scatter plot

X= train_df['X2 house age']
y = train_df['Y price sq feet']
X2= np.array(X).reshape(-1,1) # reshape(-1,1) makes the array
horizontal, y is vertical

# Train the model
model = sklearn.linear_model.LinearRegression()
model.fit(X2, y)
r_sq = model.score(X2, y)
print(r_sq)
plt.scatter(X2, y, color='red', label='observed')

plt.plot(X2, model.predict(X2), label='fit', color='Green',
linewidth=2)

plt.xlabel('X2 house age')
plt.ylabel('Y price sq feet')
plt.title('Linear Regression')
plt.legend(loc='best')

plt.show()
```

```
0.7349929890443487
```



The Selection Of Attributes For Regression

Based on the R-squared values obtained:

X2 house age: It has a high R-squared value of 0.735, explaining 73.5% of the variance in 'Y price sq feet', making it a significant predictor.

X3 bus station: With an R-squared value of 0.001, it explains only 0.1% of the variance in 'Y price sq feet', indicating weak predictive power.

X4 number of stores: Its R-squared value of 0.004 suggests it explains only 0.4% of the variance in 'Y price sq feet', making it a weak predictor as well.

A higher R-squared value indicates a better fit of the regression model to the data, suggesting that the independent variable(s) are more effective in explaining the variability in the dependent variable. In this case, an R-squared value of 0.73 for 'X2 house age' means that approximately 73% of the variance in 'Y price sq feet' is explained by the 'X2 house age' variable, indicating a strong linear relationship between them. This assessment helps in understanding the predictive performance of the linear regression model.

```
# Make a prediction
X3= test_df['X2 house age']

X3= np.array(X3).reshape(-1,1)
```

```
#predicts the price for the house age at index 40 in the X3 array
predicted_price = model.predict([X3[40]])
print('The predicted price for the specified house age using the
trained linear regression model (model).')
print('house age= {:.2f} predicted price = {:.2f}'.format(X3[40][0],
predicted_price[0]))
```

```
The predicted price for the specified house age using the trained
linear regression model (model).
house age= 35.70 predicted price = 80.27
```

Conclusion :

Data cleaning, including replacing zeros with appropriate summary statistics and handling missing values, is crucial for ensuring dataset integrity. Attribute selection for regression involves choosing variables with high predictive power. Regression scores, obtained using the model.score() function, help assess attribute performance in predicting the target variable. In this analysis, 'X2 house age' demonstrated the highest predictive ability, as indicated by its regression score. Key learnings include the importance of data preprocessing and attribute selection in regression analysis, while challenges revolved around handling outliers and selecting relevant variables.

Reflection:

This assignment reinforced the significance of data cleaning and attribute selection in regression analysis. Handling missing values and outliers, and identifying variables with strong predictive power were key challenges. Through this task, I gained a deeper understanding of data preprocessing techniques and their impact on regression model performance. Moving forward, I aim to refine my skills in feature engineering and model evaluation to enhance predictive accuracy in regression tasks.