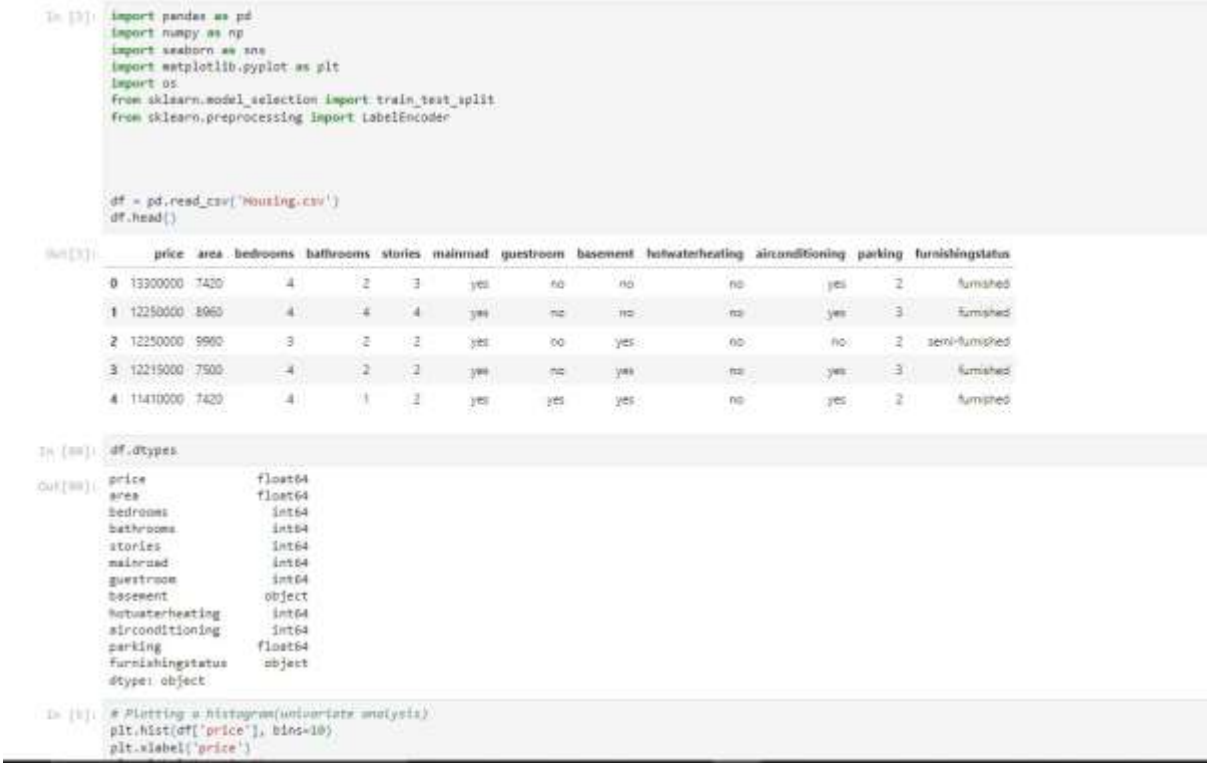
NAME: SURANA GAURAV

ROLL: 20BEE0371

ASSIGNMENT 3

Date : 06-06-2023



In [3]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder




df = pd.read_csv('Housing.csv')
df.head()
```

Out[3]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	furnished

In [88]:

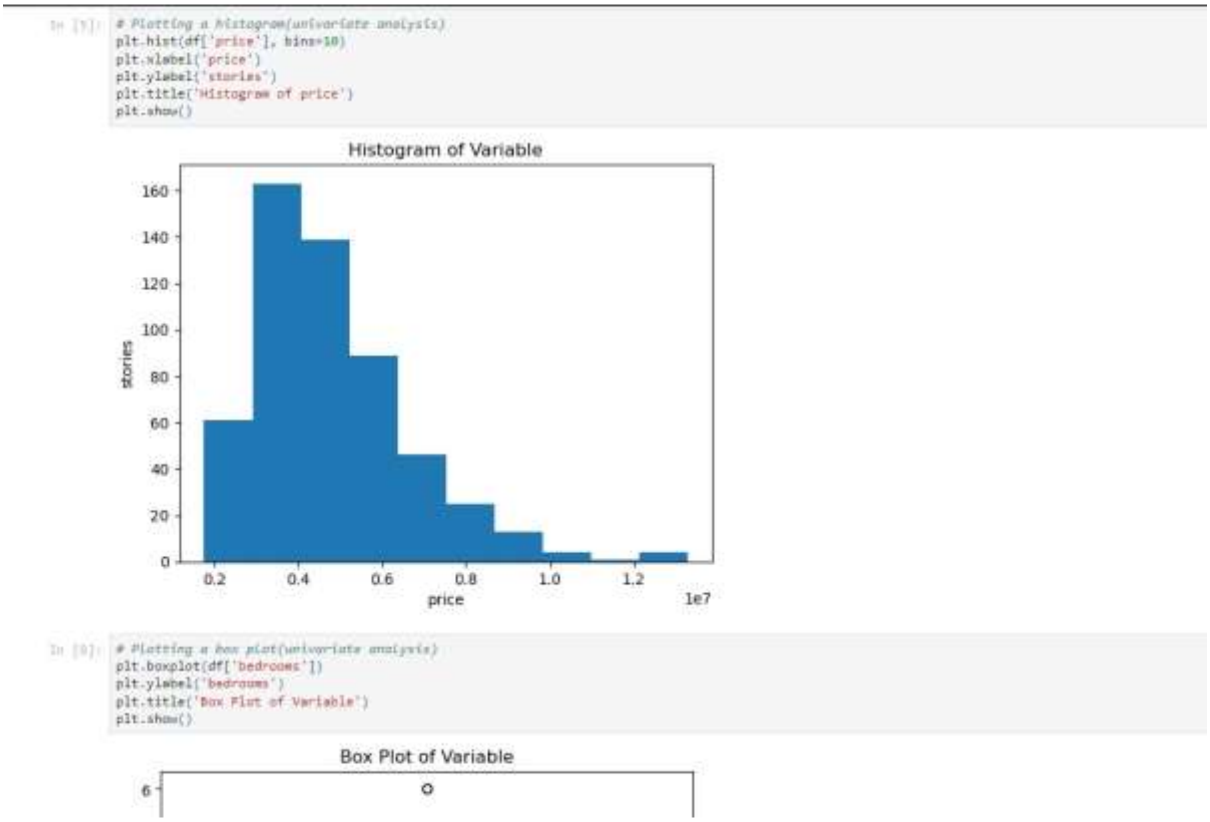
```
df.dtypes
```

Out[88]:

```
price                float64
area                 float64
bedrooms               int64
bathrooms              int64
stories                int64
mainroad               int64
guestroom              int64
basement              object
hotwaterheating        int64
airconditioning        int64
parking              float64
furnishingstatus      object
dtype: object
```

In [6]:

```
# Plotting a histogram(univariate analysis)
plt.hist(df['price'], bins=10)
plt.xlabel('price')
```

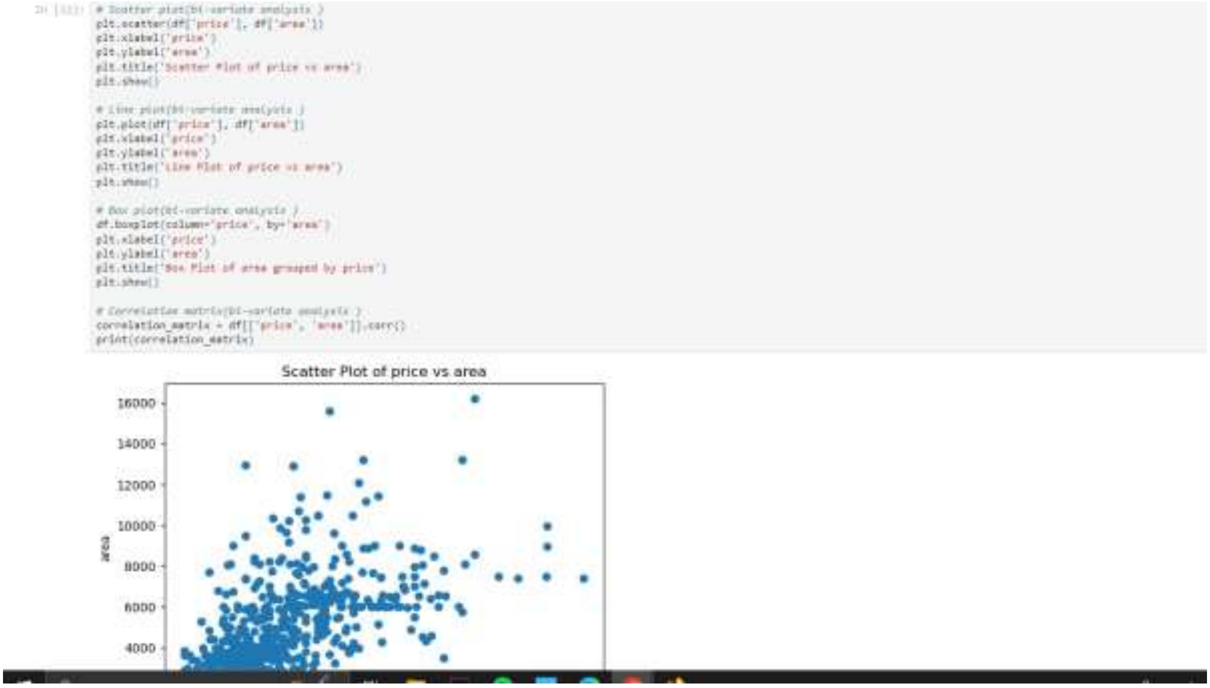


In [7]:

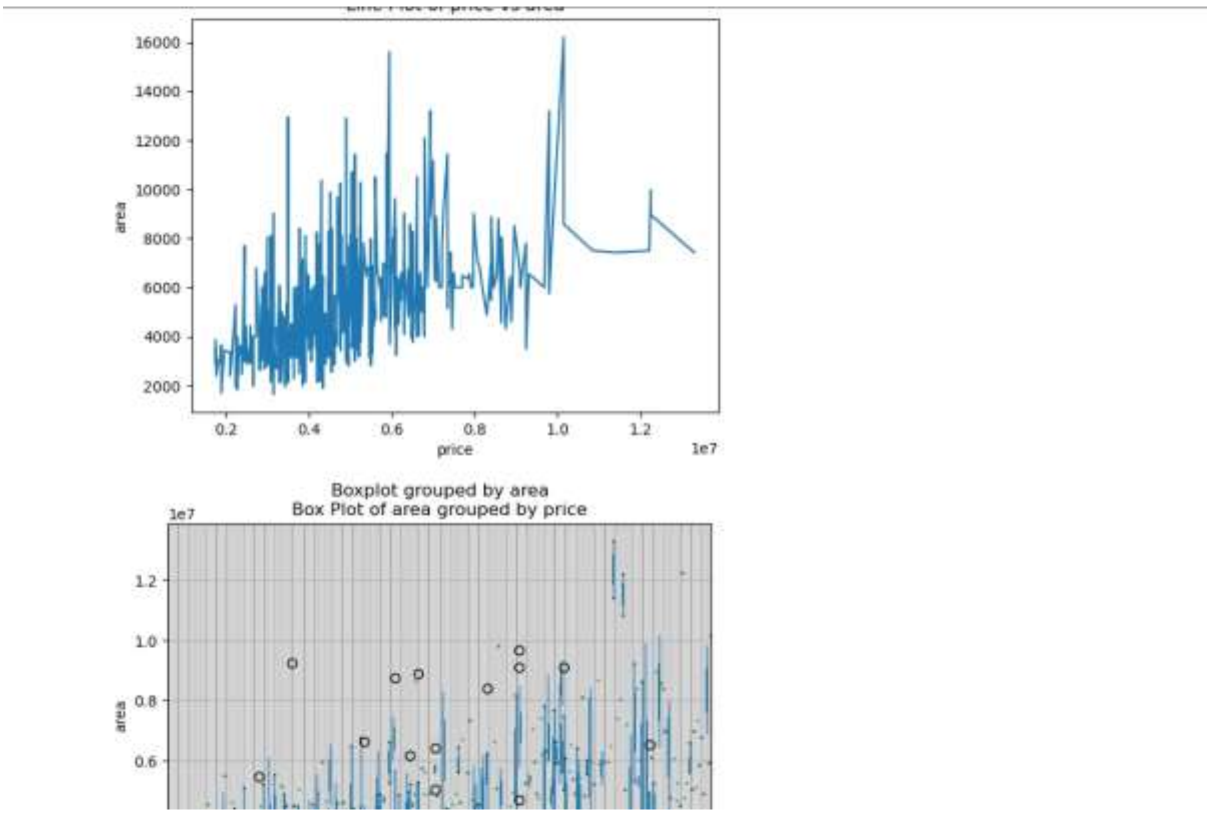
```
# Plotting a histogram(univariate analysis)
plt.hist(df['price'], bins=10)
plt.xlabel('price')
plt.ylabel('stories')
plt.title('Histogram of price')
plt.show()
```

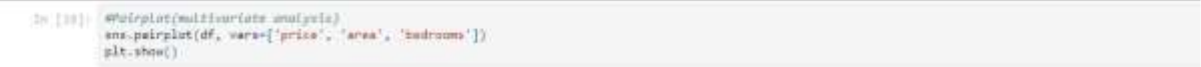
In [8]:

```
# Plotting a box plot(univariate analysis)
plt.boxplot(df['bedrooms'])
plt.ylabel('bedrooms')
plt.title('Box Plot of Variable')
plt.show()
```

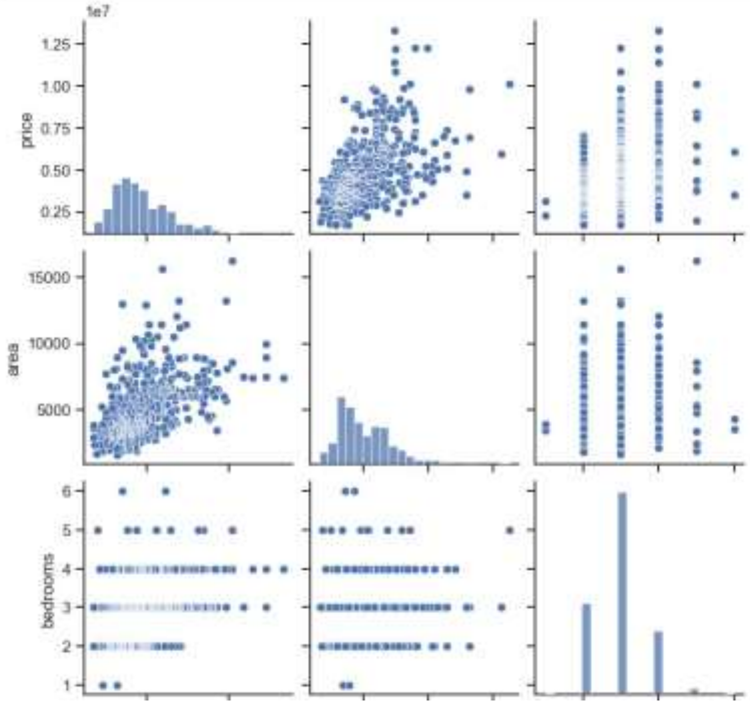


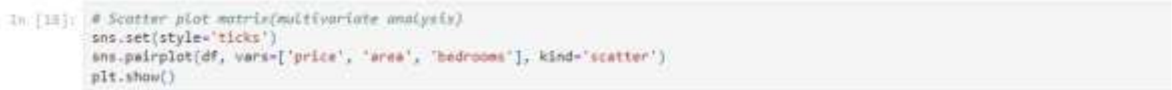
```
# Scatter plot(bi-variate analysis )
plt.scatter(df['price'], df['area'])
plt.xlabel('price')
plt.ylabel('area')
plt.title('Scatter Plot of price vs area')
plt.show()

# Line plot(bi-variate analysis )
plt.plot(df['price'], df['area'])
plt.xlabel('price')
plt.ylabel('area')
plt.title('Line Plot of price vs area')
plt.show()

# Box plot(bi-variate analysis )
df.boxplot(column='price', by='area')
plt.xlabel('price')
plt.ylabel('area')
plt.title('Box Plot of area grouped by price')
plt.show()

# Correlation matrix(bi-variate analysis )
correlation_matrix = df[['price', 'area']].corr()
print(correlation_matrix)
```



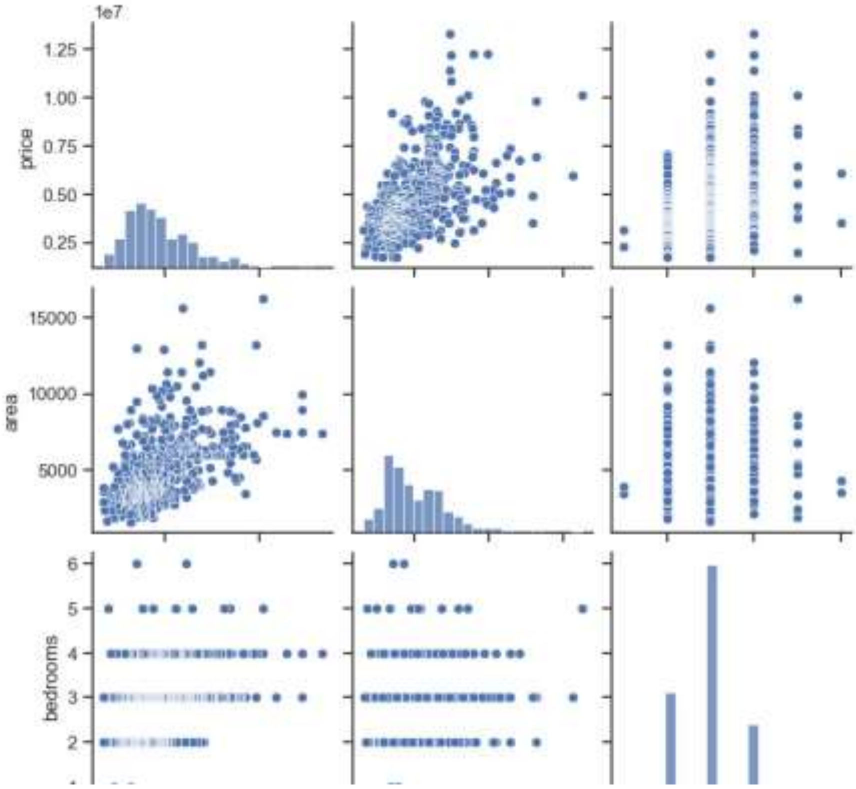


```
In [18]: #Pairplot(multivariate analysis)
         sns.pairplot(df, vars=['price', 'area', 'bedrooms'])
         plt.show()
```

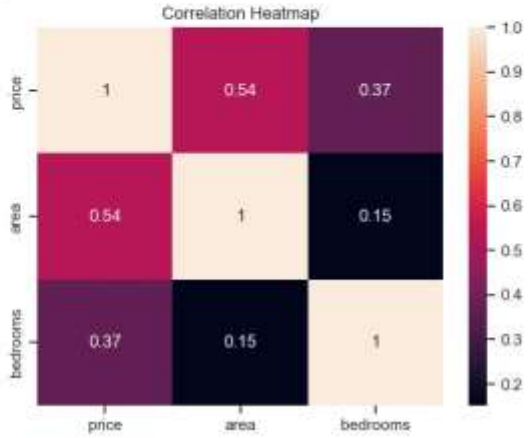




```
In [18]: # Scatter plot matrix(multivariate analysis)
         sns.set(style='ticks')
         sns.pairplot(df, vars=['price', 'area', 'bedrooms'], kind='scatter')
         plt.show()
```



```
In [17]: # Heatmap(multivariate analysis)
         correlation_matrix = df[['price', 'area', 'bedrooms']].corr()
         sns.heatmap(correlation_matrix, annot=True)
         plt.title('Correlation Heatmap')
         plt.show()
```



```
In [21]: # Get summary statistics of the entire dataset(descriptive statistics)
         print(df.describe())
```

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

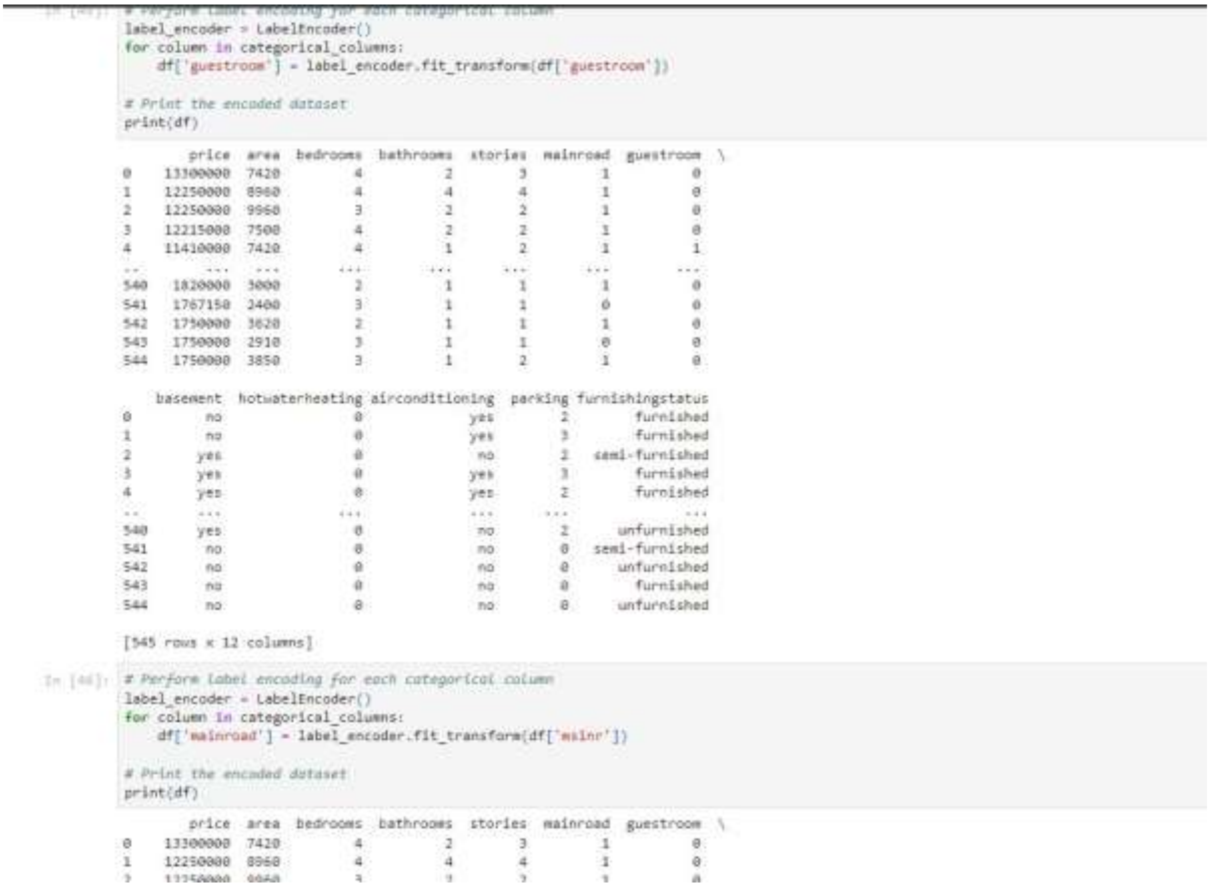
```
In [22]: # Get summary statistics for a specific column/variable(descriptive statistics)
         print(df['area'].describe())
```

```
count      545.000000
mean      5150.541284
std       2170.141023
min       1650.000000
25%       3600.000000
50%       4600.000000
75%       6360.000000
max      16200.000000
Name: area, dtype: float64
```

```
In [24]: # Check for missing values
         missing_values = df.isnull().sum()
         print(missing_values)
         #No missing values
```

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

```
In [25]: # Defining the columns for which we want to detect outliers
         columns_to_check = ['price', 'area']

         # Calculate the IQR for each column
         Q1 = df [columns_to_check].quantile(0.25)
         Q3 = df[columns_to_check].quantile(0.75)
         IQR = Q3 - Q1

         # Detecting outliers based on the IQR method
         outliers = ((df[columns_to_check] < (Q1 - 1.5 * IQR)) | (df[columns_to_check] > (Q3 + 1.5 * IQR)))

         # Replacing outliers with the median value
         df[outliers] = np.nan  # Replacing outliers with NaN
         df_filled = df.fillna(df.median())  # Replacing NaN with the median
```



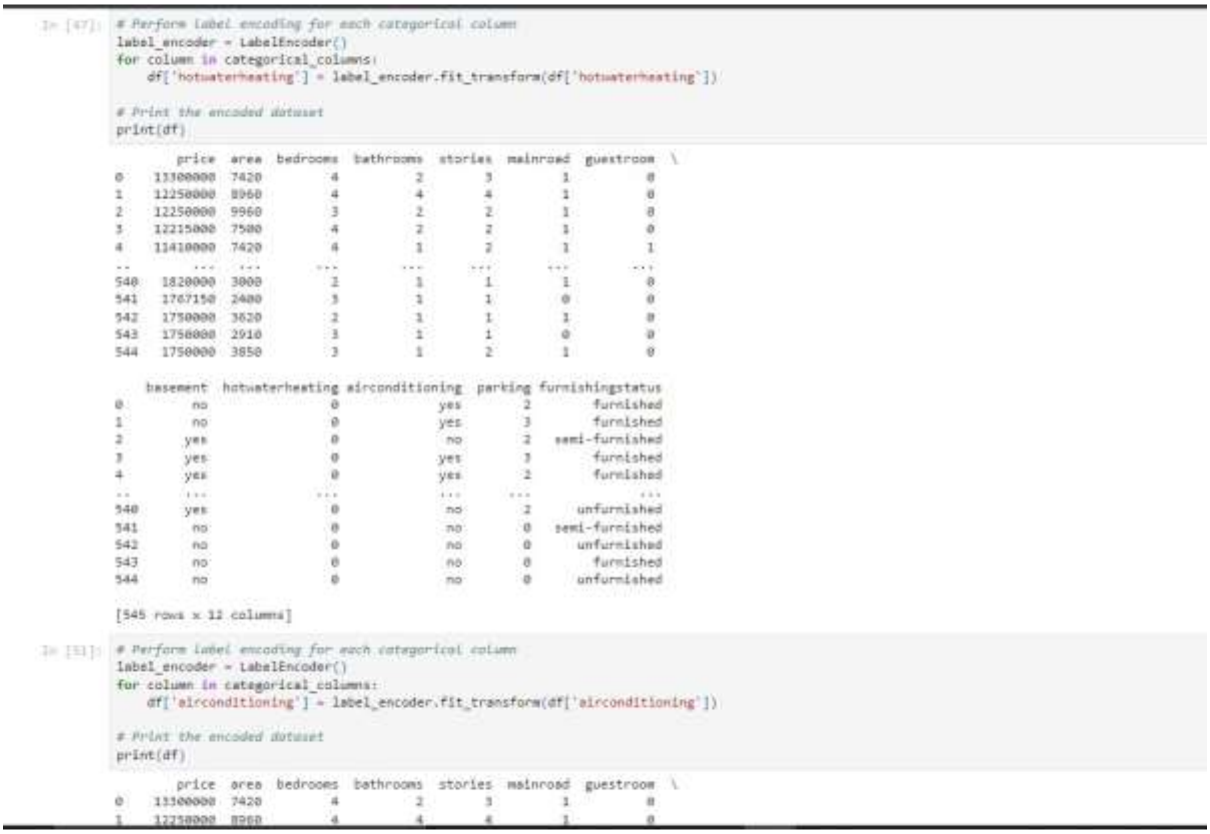
```
# Perform label encoding for each categorical column
label_encoder = LabelEncoder()
for column in categorical_columns:
    df['guestroom'] = label_encoder.fit_transform(df['guestroom'])

# Print the encoded dataset
print(df)
```

```
        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0    13300000  7420         4          2        3         1          0   
1    12250000  8960         4          4        4         1          0   
2    12250000  9960         3          2        2         1          0   
3    12215000  7500         4          2        2         1          0   
4    11410000  7420         4          1        2         1          1   
..        ...   ...       ...        ...      ...       ...        ...   
540   1820000  3000         2          1        1         1          0   
541   1767150  2400         3          1        1         0          0   
542   1750000  3620         2          1        1         1          0   
543   1750000  2910         3          1        1         0          0   
544   1750000  3850         3          1        2         1          0   

    basement  hotwaterheating airconditioning  parking furnishingstatus  
0         no                0             yes        2        furnished  
1         no                0             yes        3        furnished  
2        yes                0              no        2   semi-furnished  
3        yes                0             yes        3        furnished  
4        yes                0             yes        2        furnished  
..       ...              ...             ...      ...              ...  
540      yes                0              no        2      unfurnished  
541       no                0              no        0   semi-furnished  
542       no                0              no        0      unfurnished  
543       no                0              no        0        furnished  
544       no                0              no        0      unfurnished  

[545 rows x 12 columns]
```

In [46]:

```
# Perform label encoding for each categorical column
label_encoder = LabelEncoder()
for column in categorical_columns:
    df['mainroad'] = label_encoder.fit_transform(df['mainr'])

# Print the encoded dataset
print(df)
```

```
        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0    13300000  7420         4          2        3         1          0   
1    12250000  8960         4          4        4         1          0   
2    12250000  9960         3          2        2         1          0   
```



In [47]:

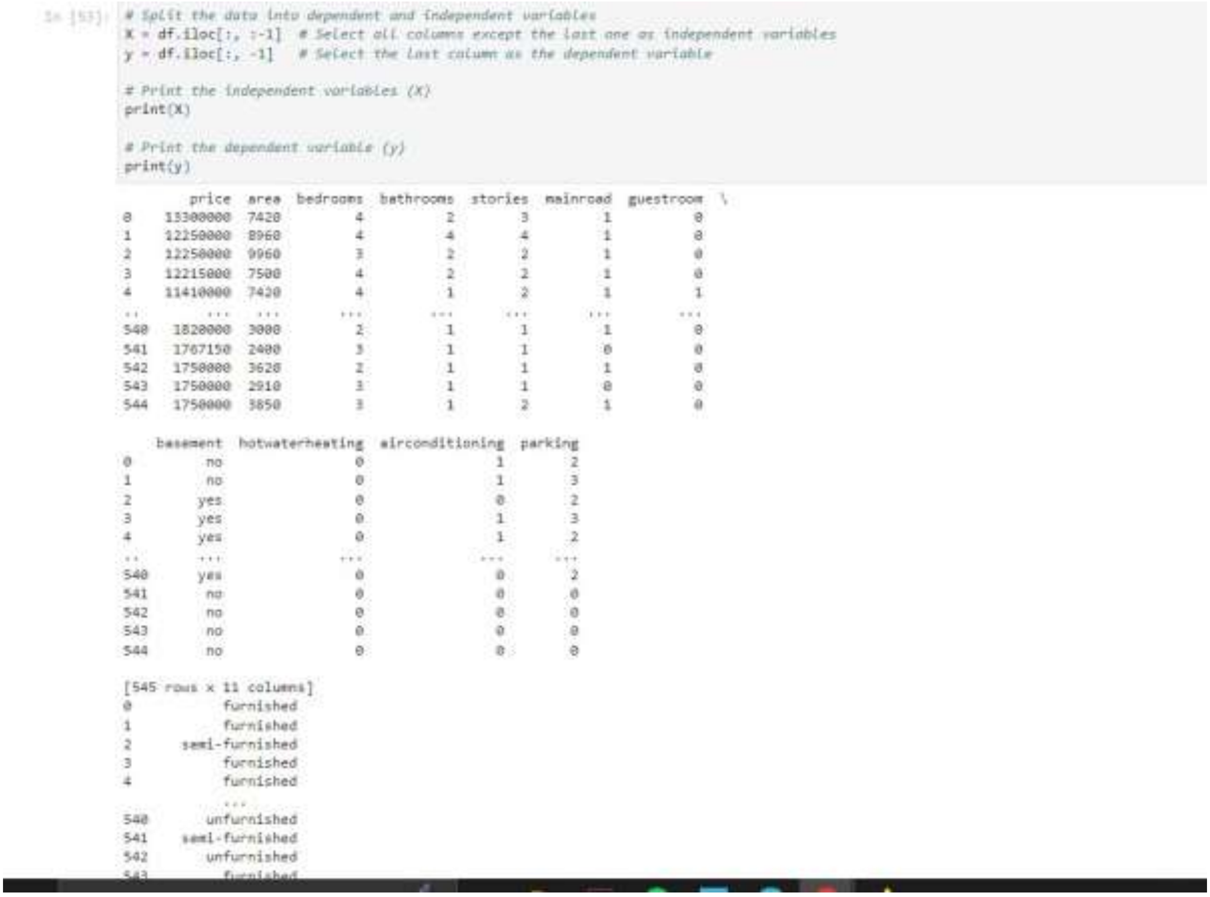
```
# Perform label encoding for each categorical column
label_encoder = LabelEncoder()
for column in categorical_columns:
    df['hotwaterheating'] = label_encoder.fit_transform(df['hotwaterheating'])

# Print the encoded dataset
print(df)
```

```
        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0    13300000  7420         4          2        3         1          0   
1    12250000  8960         4          4        4         1          0   
2    12250000  9960         3          2        2         1          0   
3    12215000  7500         4          2        2         1          0   
4    11410000  7420         4          1        2         1          1   
..        ...   ...       ...        ...      ...       ...        ...   
540   1820000  3000         2          1        1         1          0   
541   1767150  2400         3          1        1         0          0   
542   1750000  3620         2          1        1         1          0   
543   1750000  2910         3          1        1         0          0   
544   1750000  3850         3          1        2         1          0   

    basement  hotwaterheating airconditioning  parking furnishingstatus  
0         no                0             yes        2        furnished  
1         no                0             yes        3        furnished  
2        yes                0              no        2   semi-furnished  
3        yes                0             yes        3        furnished  
4        yes                0             yes        2        furnished  
..       ...              ...             ...      ...              ...  
540      yes                0              no        2      unfurnished  
541       no                0              no        0   semi-furnished  
542       no                0              no        0      unfurnished  
543       no                0              no        0        furnished  
544       no                0              no        0      unfurnished  

[545 rows x 12 columns]
```

In [51]:

```
# Perform label encoding for each categorical column
label_encoder = LabelEncoder()
for column in categorical_columns:
    df['airconditioning'] = label_encoder.fit_transform(df['airconditioning'])

# Print the encoded dataset
print(df)
```

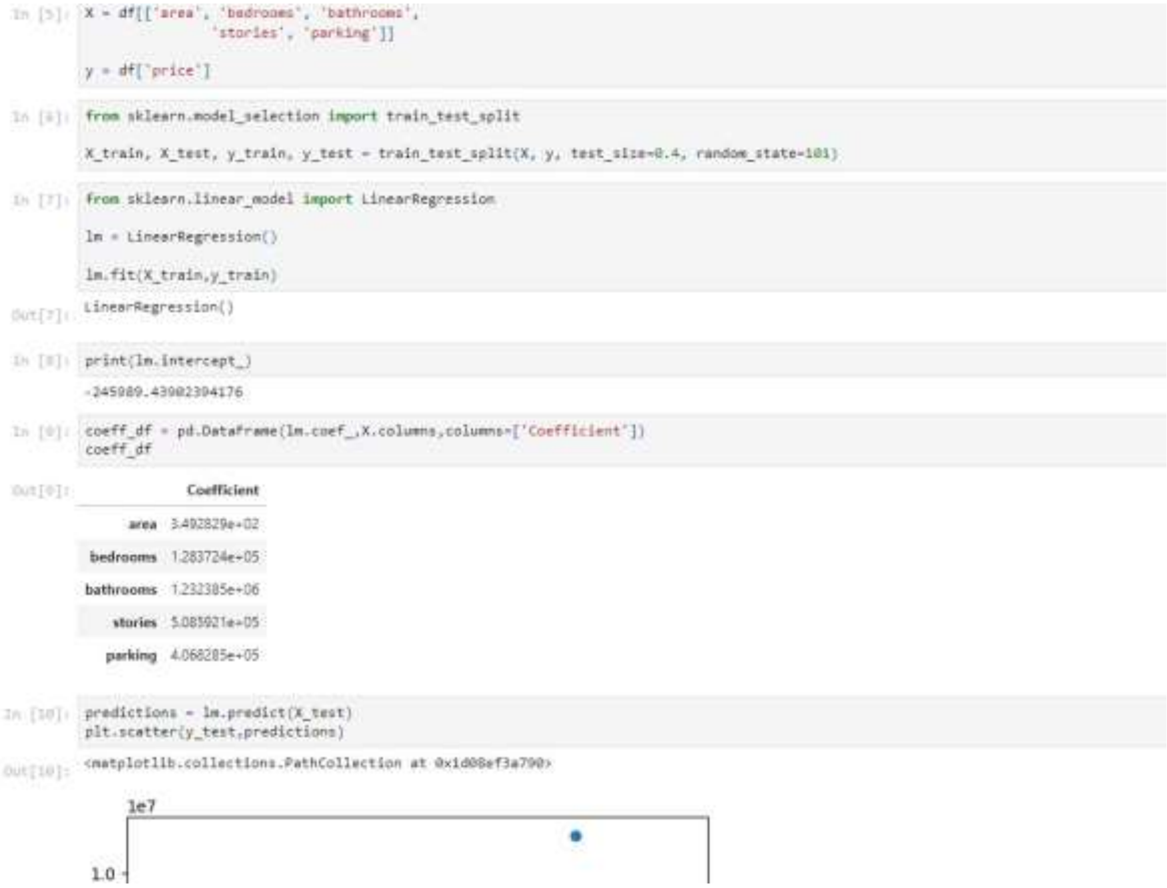
```
        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0    13300000  7420         4          2        3         1          0   
1    12250000  8960         4          4        4         1          0   
```



In [53]:

```
# Split the data into dependent and independent variables
X = df.iloc[:, :-1]  # Select all columns except the last one as independent variables
y = df.iloc[:, -1]   # Select the last column as the dependent variable

# Print the independent variables (X)
print(X)

# Print the dependent variable (y)
print(y)
```

```
        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0    13300000  7420         4          2        3         1          0
1    12250000  8960         4          4        4         1          0
2    12250000  9960         3          2        2         1          0
3    12215000  7500         4          2        2         1          0
4    11410000  7420         4          1        2         1          1
..        ...   ...       ...        ...      ...       ...        ...
540   1820000  3000         2          1        1         1          0
541   1767150  2400         3          1        1         0          0
542   1750000  3620         2          1        1         1          0
543   1750000  2910         3          1        1         0          0
544   1750000  3850         3          1        2         1          0

    basement  hotwaterheating  airconditioning  parking
0         no                0                1        2
1         no                0                1        3
2        yes                0                0        2
3        yes                0                1        3
4        yes                0                1        2
..       ...              ...              ...      ...
540      yes                0                0        2
541       no                0                0        0
542       no                0                0        0
543       no                0                0        0
544       no                0                0        0

[545 rows x 11 columns]
0           furnished
1           furnished
2      semi-furnished
3           furnished
4           furnished
            ...
540       unfurnished
541    semi-furnished
542       unfurnished
543         furnished
```



In [5]:

```
X = df[['area', 'bedrooms', 'bathrooms',
               'stories', 'parking']]

y = df['price']
```

In [6]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

In [7]:

```
from sklearn.linear_model import LinearRegression

lm = LinearRegression()

lm.fit(X_train,y_train)
```

Out[7]:

```
LinearRegression()
```

In [8]:

```
print(lm.intercept_)
```

```
-245989.43902394176
```

In [9]:

```
coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])
coeff_df
```

Out[9]:

	Coefficient
area	3.492829e+02
bedrooms	1.283724e+05
bathrooms	1.232385e+06
stories	5.085921e+05
parking	4.068285e+05

In [10]:

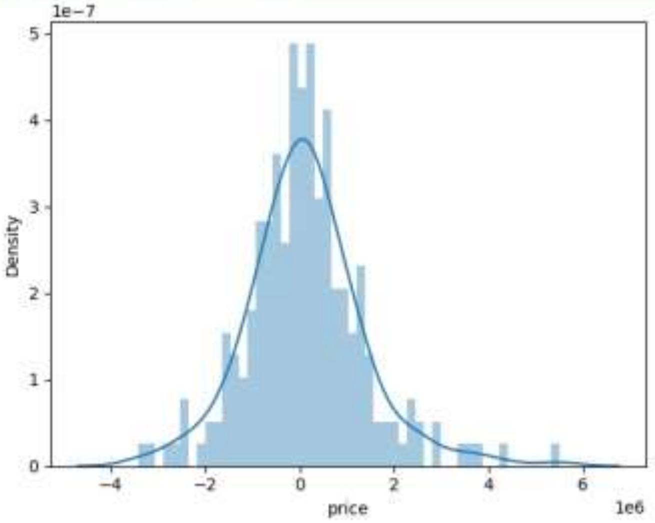
```
predictions = lm.predict(X_test)
plt.scatter(y_test,predictions)
```

Out[10]:

```
<matplotlib.collections.PathCollection at 0x1d08ef3a790>
```

```
In [11]: sns.distplot((y_test-predictions),bins=50);
```

```
C:\Users\lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function
`displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



```
In [ ]: from sklearn import metrics

        mse = mean_squared_error(y_test, predictions)
        print('Mean Squared Error:', mse)
        Mean Squared Error: 0.14268744138998152
```