```
#python
import warnings
warnings.filterwarnings('ignore')

# Import the numpy and pandas package

import numpy as np
import pandas as pd

# Data Visualisation

import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Correct URL to the raw CSV data
url = "https://raw.githubusercontent.com/satyanarayanan102/OIBSIP/main/Housing%20Price%20Prediction/Housing%20(1).csv"
housing = pd.DataFrame(pd.read_csv(url))
```

```
housing.head()
```

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

```
housing.shape
```

```
(545, 13)
```

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

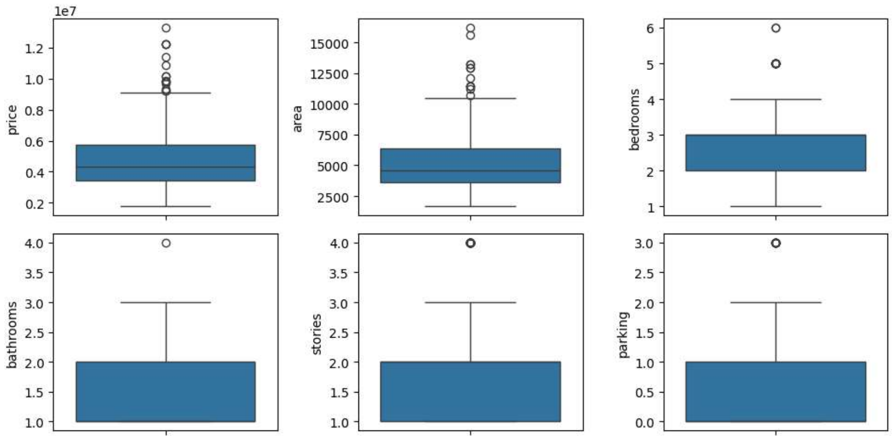
```
housing.describe()
```

	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

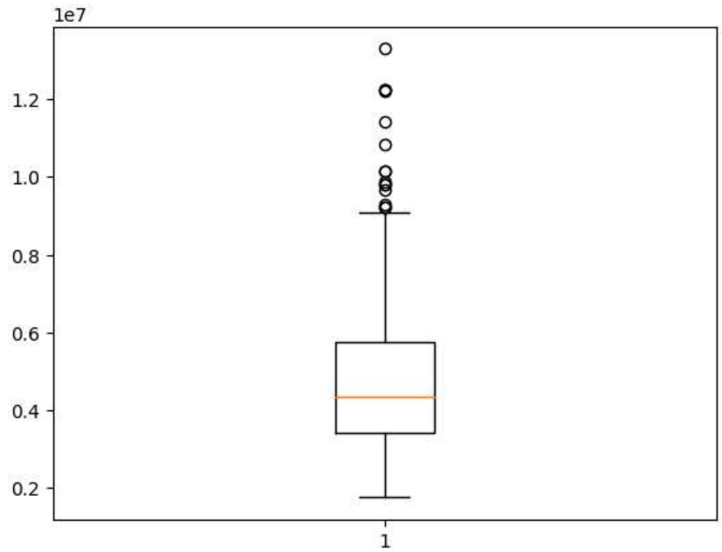
```
# Checking Null values
housing.isnull().sum()*100/housing.shape[0]
```

```
price               0.0
area                0.0
bedrooms            0.0
bathrooms           0.0
stories             0.0
mainroad            0.0
guestroom           0.0
basement            0.0
hotwaterheating     0.0
airconditioning     0.0
parking             0.0
prefarea            0.0
furnishingstatus    0.0
dtype: float64
```

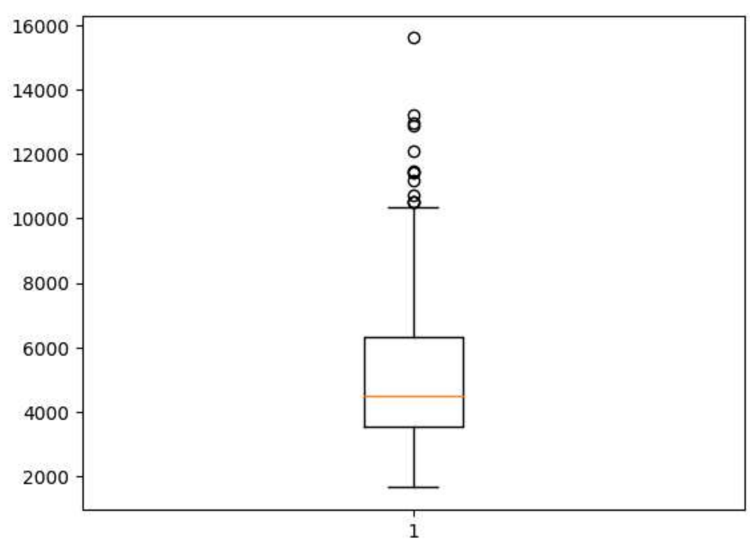
```
# Outlier Analysis
fig, axs = plt.subplots(2,3, figsize = (10,5))
plt1 = sns.boxplot(housing['price'], ax = axs[0,0])
plt2 = sns.boxplot(housing['area'], ax = axs[0,1])
plt3 = sns.boxplot(housing['bedrooms'], ax = axs[0,2])
plt1 = sns.boxplot(housing['bathrooms'], ax = axs[1,0])
plt2 = sns.boxplot(housing['stories'], ax = axs[1,1])
plt3 = sns.boxplot(housing['parking'], ax = axs[1,2])
plt.tight_layout()
```



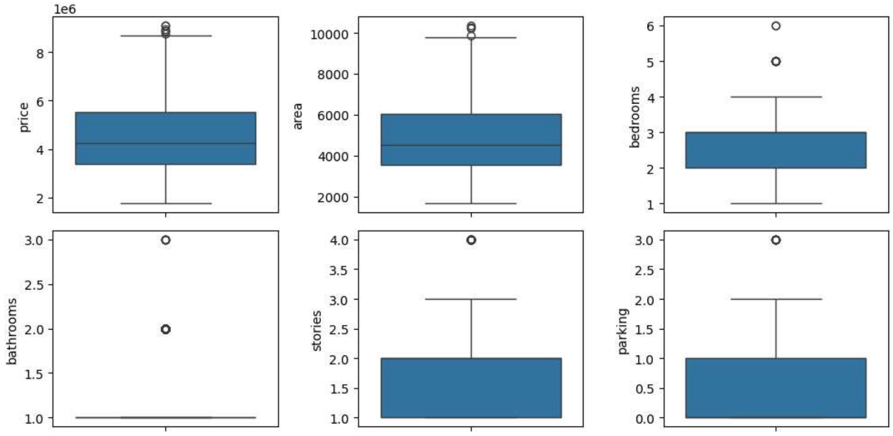
```
# outlier treatment for price
plt.boxplot(housing.price)
Q1 = housing.price.quantile(0.25)
Q3 = housing.price.quantile(0.75)
IQR = Q3 - Q1
housing = housing[(housing.price >= Q1 - 1.5*IQR) & (housing.price <= Q3 + 1.5*IQR)]
```



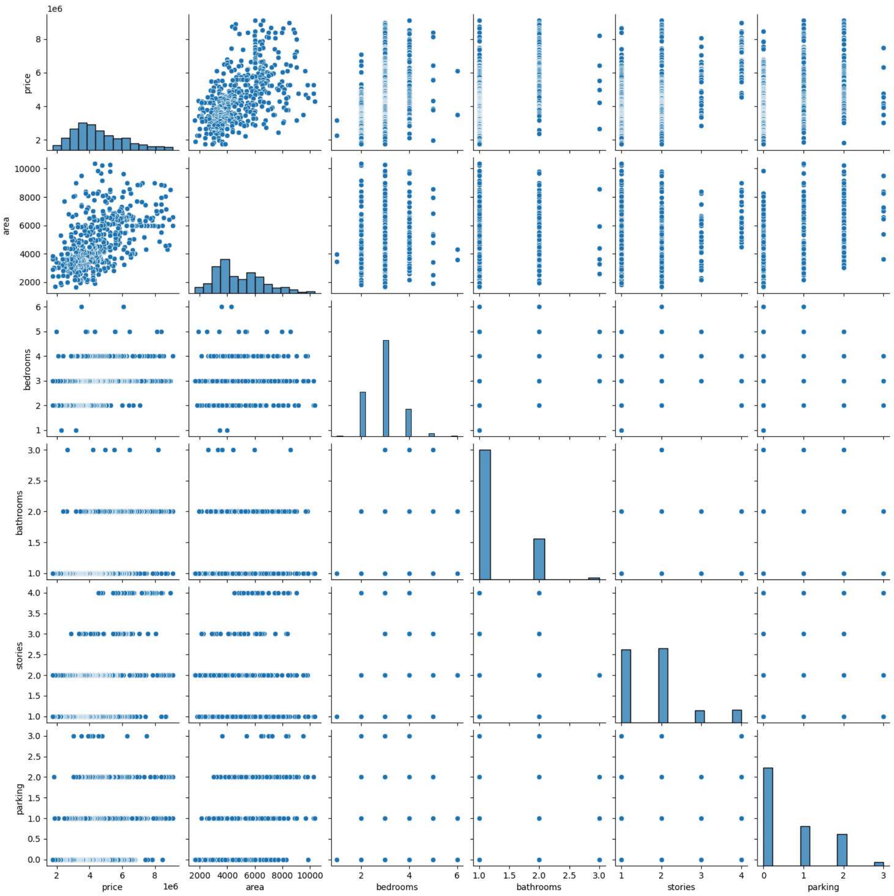

```
# outlier treatment for area
plt.boxplot(housing.area)
Q1 = housing.area.quantile(0.25)
Q3 = housing.area.quantile(0.75)
IQR = Q3 - Q1
housing = housing[(housing.area >= Q1 - 1.5*IQR) & (housing.area <= Q3 + 1.5*IQR)]
```



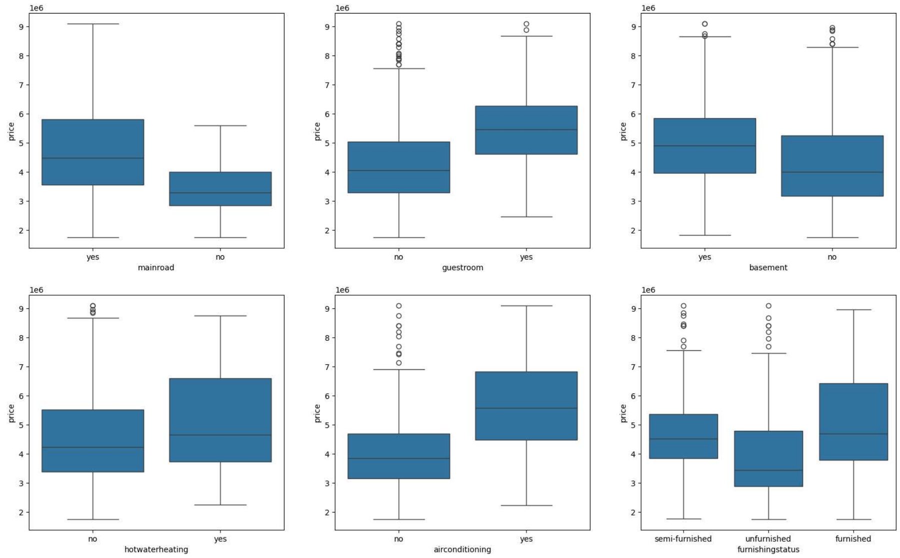

```
# Outlier Analysis
fig, axs = plt.subplots(2,3, figsize = (10,5))
plt1 = sns.boxplot(housing['price'], ax = axs[0,0])
plt2 = sns.boxplot(housing['area'], ax = axs[0,1])
plt3 = sns.boxplot(housing['bedrooms'], ax = axs[0,2])
plt1 = sns.boxplot(housing['bathrooms'], ax = axs[1,0])
plt2 = sns.boxplot(housing['stories'], ax = axs[1,1])
plt3 = sns.boxplot(housing['parking'], ax = axs[1,2])

plt.tight_layout()
```



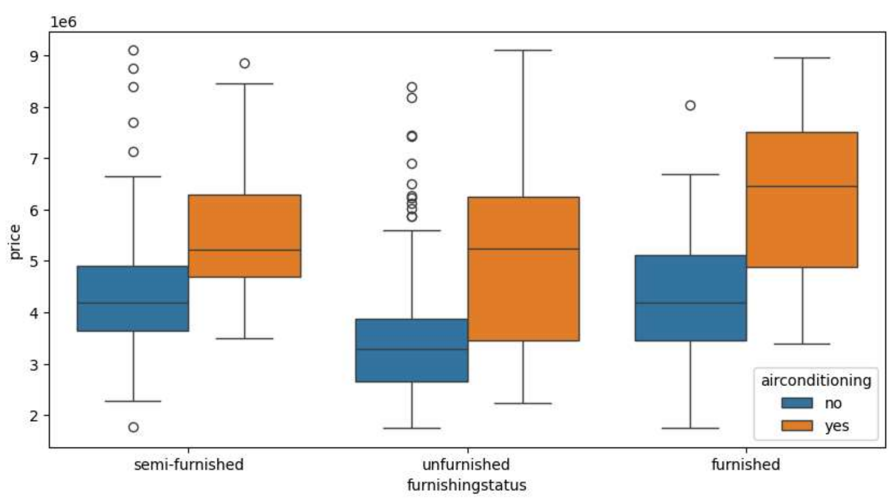

```
sns.pairplot(housing)
plt.show()
```




```
plt.figure(figsize=(20, 12))
plt.subplot(2,3,1)
sns.boxplot(x = 'mainroad', y = 'price', data = housing)
plt.subplot(2,3,2)
sns.boxplot(x = 'guestroom', y = 'price', data = housing)
plt.subplot(2,3,3)
sns.boxplot(x = 'basement', y = 'price', data = housing)
plt.subplot(2,3,4)
sns.boxplot(x = 'hotwaterheating', y = 'price', data = housing)
plt.subplot(2,3,5)
sns.boxplot(x = 'airconditioning', y = 'price', data = housing)
plt.subplot(2,3,6)
sns.boxplot(x = 'furnishingstatus', y = 'price', data = housing)
plt.show()
```




```
plt.figure(figsize = (10, 5))
sns.boxplot(x = 'furnishingstatus', y = 'price', hue = 'airconditioning', data = housing)
plt.show()
```



```
varlist =  ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea']

# Defining the map function
def binary_map(x):
    return x.map({'yes': 1, "no": 0})

# Applying the function to the housing list
housing[varlist] = housing[varlist].apply(binary_map)
```

```
housing.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwater
15	9100000	6000	4	1	2	1	0	1	
16	9100000	6600	4	2	2	1	1	1	
17	8960000	8500	3	2	4	1	0	0	
18	8890000	4600	3	2	2	1	1	0	
19	8855000	6420	3	2	2	1	0	0	

```
# Get the dummy variables for the feature 'furnishingstatus' and store it in a new variable - 'status'
status = pd.get_dummies(housing['furnishingstatus'])
# Check what the dataset 'status' looks like
status.head()
```

	furnished	semi-furnished	unfurnished
15	False	True	False
16	False	False	True
17	True	False	False
18	True	False	False
19	False	True	False

```
status = pd.get_dummies(housing['furnishingstatus'], drop_first = True)
# Add the results to original housing dataframe

housing = pd.concat([housing, status], axis = 1)

housing.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwater
15	9100000	6000	4	1	2	1	0	1	
16	9100000	6600	4	2	2	1	1	1	
17	8960000	8500	3	2	4	1	0	0	
18	8890000	4600	3	2	2	1	1	0	

```
housing.drop(['furnishingstatus'], axis = 1, inplace = True)
housing.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwater
15	9100000	6000	4	1	2	1	0	1	
16	9100000	6600	4	2	2	1	1	1	
17	8960000	8500	3	2	4	1	0	0	
18	8890000	4600	3	2	2	1	1	0	

```
from sklearn.model_selection import train_test_split

np.random.seed(0)
df_train, df_test = train_test_split(housing, train_size = 0.7, test_size = 0.3, random_state = 100)
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

```
num_vars = ['area', 'bedrooms', 'bathrooms', 'stories', 'parking','price']

df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
```

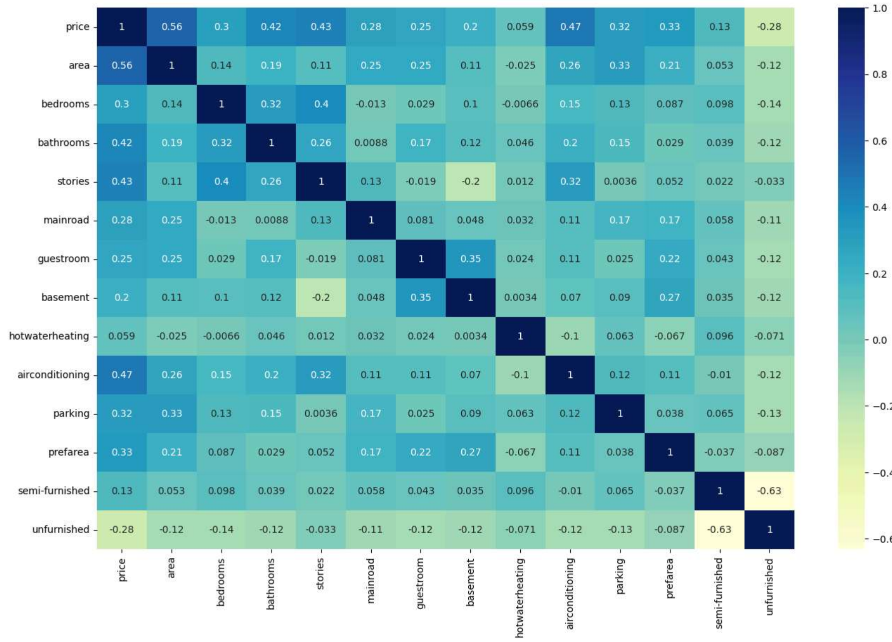
```
df_train.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	ho
148	0.523810	0.526907	0.4	0.0	0.666667	1	0	0	
236	0.390476	0.114134	0.2	0.0	0.333333	1	1	1	
356	0.275238	0.072738	0.8	0.5	0.000000	0	0	1	
425	0.219048	0.151390	0.2	0.0	0.000000	1	0	1	

```
df_train.describe()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom
count	361.000000	361.000000	361.000000	361.000000	361.000000	361.000000	361.000000
mean	0.383701	0.350081	0.390582	0.127424	0.268698	0.875346	0.168975
std	0.209712	0.207184	0.149146	0.224465	0.287833	0.330784	0.375250
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.237143	0.189829	0.200000	0.000000	0.000000	1.000000	0.000000
50%	0.338095	0.295092	0.400000	0.000000	0.333333	1.000000	0.000000
75%	0.514286	0.491425	0.400000	0.000000	0.333333	1.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
plt.figure(figsize = (16, 10))
sns.heatmap(df_train.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



```
y_train = df_train.pop('price')
X_train = df_train
```

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
# Running RFE with the output number of the variable equal to 10
lm = LinearRegression()
lm.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
rfe = RFE(lm, n_features_to_select=6)
```

```
rfe = rfe.fit(X_train, y_train)
```

```
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

```
[('area', True, 1),
 ('bedrooms', False, 7),
 ('bathrooms', True, 1),
 ('stories', True, 1),
 ('mainroad', False, 5),
```

```
 ('guestroom', False, 6),
 ('basement', False, 4),
 ('hotwaterheating', False, 2),
 ('airconditioning', True, 1),
 ('parking', True, 1),
 ('prefarea', True, 1),
 ('semi-furnished', False, 8),
 ('unfurnished', False, 3)]
```

```
col = X_train.columns[rfe.support_]
col
```

```
Index(['area', 'bathrooms', 'stories', 'airconditioning', 'parking',
       'prefarea'],
      dtype='object')
```

```
X_train.columns[~rfe.support_]
```

```
Index(['bedrooms', 'mainroad', 'guestroom', 'basement', 'hotwaterheating',
       'semi-furnished', 'unfurnished'],
      dtype='object')
```

```
X_train_rfe = X_train[col]
```

```
import statsmodels.api as sm
X_train_rfe = sm.add_constant(X_train_rfe)
```

```
lm = sm.OLS(y_train,X_train_rfe).fit()
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.611
Model:                            OLS   Adj. R-squared:                  0.605
Method:                 Least Squares   F-statistic:                     92.83
Date:                Fri, 28 Jun 2024   Prob (F-statistic):           1.31e-69
Time:                        11:13:35   Log-Likelihood:                 222.77
No. Observations:                 361   AIC:                            -431.5
Df Residuals:                     354   BIC:                            -404.3
Df Model:                           6
Covariance Type:            nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
const               0.1097      0.015      7.442      0.000       0.081       0.139
area                0.3502      0.037      9.361      0.000       0.277       0.424
bathrooms           0.2012      0.033      6.134      0.000       0.137       0.266
stories             0.1884      0.026      7.219      0.000       0.137       0.240
airconditioning     0.0965      0.016      5.890      0.000       0.064       0.129
parking             0.1009      0.026      3.916      0.000       0.050       0.152
prefarea            0.1102      0.018      6.288      0.000       0.076       0.145
==============================================================================
Omnibus:                       54.330   Durbin-Watson:                   2.060
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              125.403
Skew:                           0.762   Prob(JB):                     5.87e-28
Kurtosis:                       5.453   Cond. No.                         6.98
==============================================================================

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

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
X = X_train_rfe
vif['Features'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

	Features	VIF
0	const	4.51
1	area	1.24
4	airconditioning	1.20

```
y_train_price = lm.predict(X_train_rfe)
res = (y_train_price - y_train)
```

```
# Importing the required libraries for plots.
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# Plot the histogram
fig = plt.figure()
sns.distplot((y_train - y_train_price), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)
plt.xlabel('Errors', fontsize = 18)
```

```
Text(0.5, 0, 'Errors')
```