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

import numpy as np
import pandas as pd

housing = pd.read_csv("Housing.csv")

housing.head()
```

```
      price  area  bedrooms  bathrooms  stories mainroad guestroom
basement  \
0  13300000  7420         4          2        3      yes         no
no
1  12250000  8960         4          4        4      yes         no
no
2  12250000  9960         3          2        2      yes         no
yes
3  12215000  7500         4          2        2      yes         no
yes
4  11410000  7420         4          1        2      yes        yes
yes

  hotwaterheating airconditioning  parking prefarea furnishingstatus
0              no              yes        2      yes        furnished
1              no              yes        3       no        furnished
2              no               no        2      yes   semi-furnished
3              no              yes        3      yes        furnished
4              no              yes        2       no        furnished
```

```
housing.shape
```

```
(545, 13)
```

```
housing.info()
```

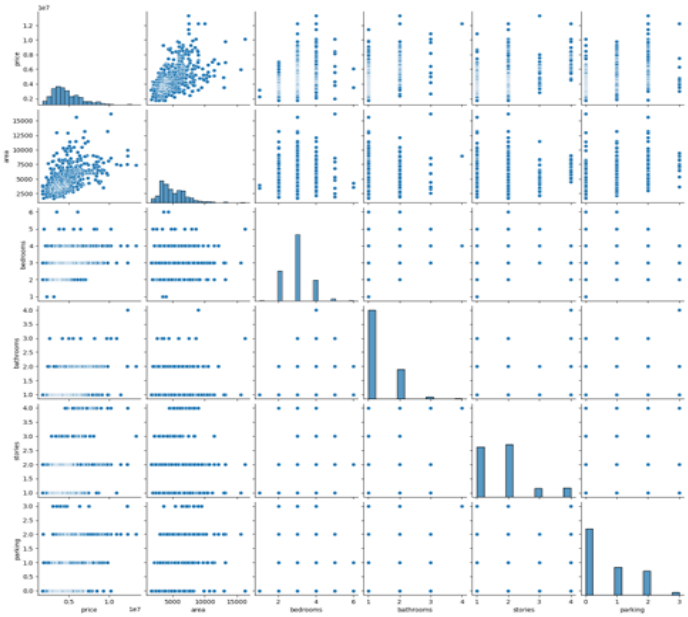
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          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
\
count  5.450000e+02    545.000000  545.000000  545.000000  545.000000

mean   4.766729e+06   5150.541284    2.965138    1.286239    1.805505

std    1.870440e+06   2170.141023    0.738064    0.502470    0.867492

min    1.750000e+06   1650.000000    1.000000    1.000000    1.000000

25%    3.430000e+06   3600.000000    2.000000    1.000000    1.000000

50%    4.340000e+06   4600.000000    3.000000    1.000000    2.000000

75%    5.740000e+06   6360.000000    3.000000    2.000000    2.000000

max    1.330000e+07  16200.000000    6.000000    4.000000    4.000000
```

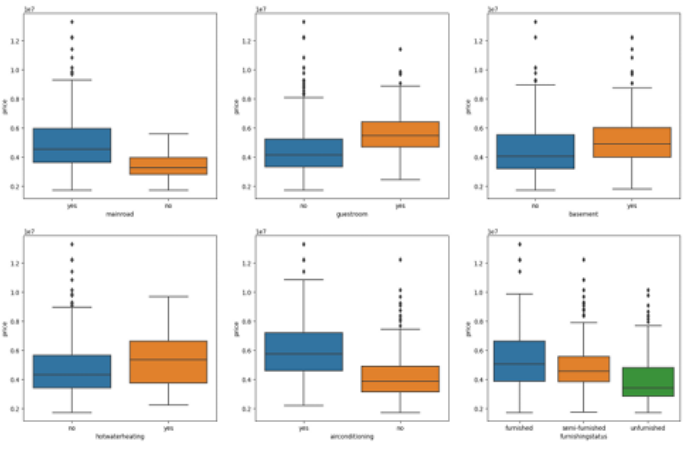
```
          parking
count  545.000000
mean     0.693578
std      0.861586
min      0.000000
25%      0.000000
50%      0.000000
75%      1.000000
max      3.000000
```

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

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

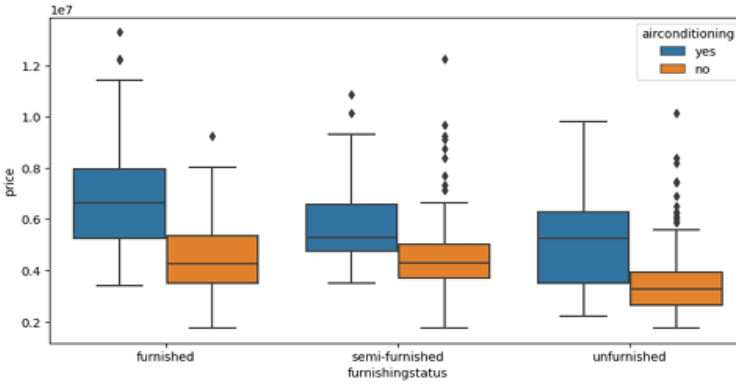


```
plt.figure(figsize=(20, 12))
plt.subplot(2,3,1)
sns.boxplot(x = 'mainroad', y = 'price', data = housing)
plt.subplot(2,3,2)
sns.boxplot(x = 'guestroom', y = 'price', data = housing)
plt.subplot(2,3,3)
sns.boxplot(x = 'basement', y = 'price', data = housing)
plt.subplot(2,3,4)
sns.boxplot(x = 'hotwaterheating',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

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

housing[varlist] = housing[varlist].apply(binary_map)

housing.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea
0	13300000	7420	4	2	3	1	0	0	0	1	2	1
1	12250000	8960	4	4	4	1	0	0	0	1	3	0
2	12250000	9960	3	2	2	1	0	1	0	0	2	1
3	12215000	7500	4	2	2	1	0	1	0	1	3	1
4	11410000	7420	4	1	2	1	1	1	0	1	2	0

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

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea
0	13300000	7420	4	2	3	1	0	0	0	1	2	1
1	12250000	8960	4	4	4	1	0	0	0	1	3	0
2	12250000	9960	3	2	2	1	0	1	0	0	2	1
3	12215000	7500	4	2	2	1	0	1	0	1	3	1
4	11410000	7420	4	1	2	1	1	1	0	1	2	0

	semi-furnished	unfurnished
0	0	0
1	0	0
2	1	0
3	0	0
4	0	0

```
from sklearn.model selection import train test split
#  We specify this so that the train and test data set always have the

# same rows, respectively
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	hotwaterheating	airconditioning	parking	prefarea
359	0.169697	0.155227	0.4	0.0	0.000000	1	0	0	0	0	0.333333	0
19	0.321212	0.115628	0.4	0.5	0.000000	1	1	0	0	1	0.333333	1
159	0.548133	0.454417	0.4	0.5	1.000000	1	0	0	0	1	0.000000	0
35	0.575758	0.538015	0.8	0.5	0.333333	1	0	0	0	1	0.666667	0
28	0.575758	0.538015	0.8	0.5	0.333333	1	0	0	0	1	0.666667	0

	furnishingstatus
0	furnished
1	furnished
2	semi-furnished
3	furnished
4	furnished

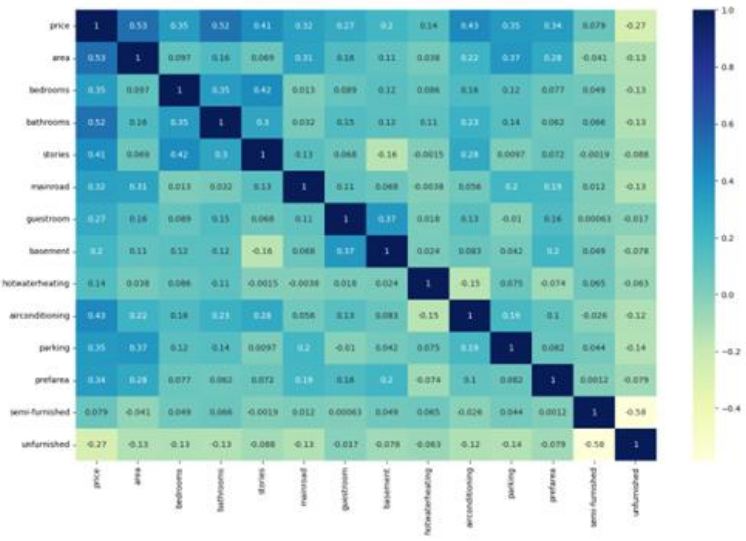
```
status = pd.get_dummies(housing['furnishingstatus'])

status.head()
```

	furnished	semi-furnished	unfurnished
0	1	0	0
1	1	0	0
2	0	1	0
3	1	0	0
4	1	0	0

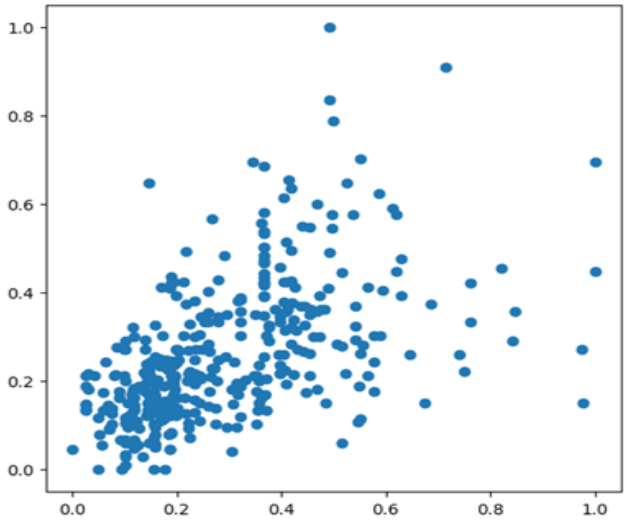
```
status = pd.get_dummies(housing['furnishingstatus'], drop_first =True)

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

housing.head()
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus	semi-furnished	unfurnished
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	furnished	0	0
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	furnished	0	0
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	semi-furnished	1	0
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	furnished	0	0
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	furnished	0	0

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



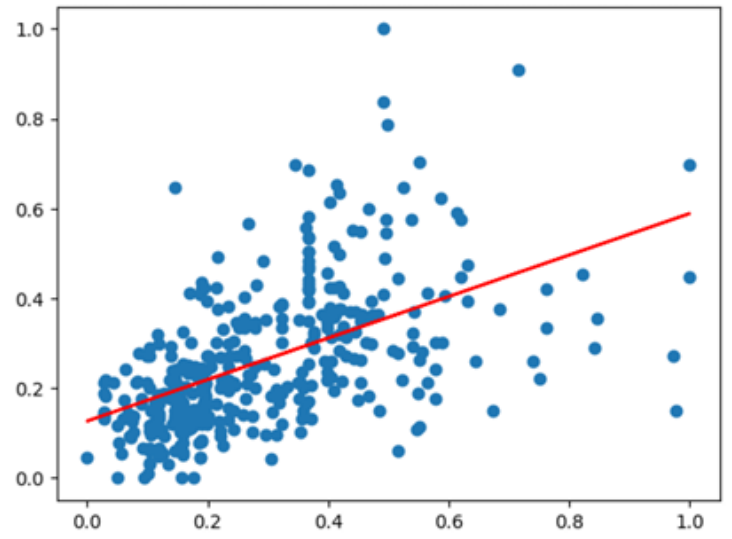
```
plt.figure(figsize=[6,6])
plt.scatter(df_train.area, df_train.price)
plt.show()
```



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

import statsmodels.api as sm
X_train_lm = sm.add_constant(X_train[['area']])
# Create a first fitted model
lr = sm.OLS(y_train, X_train_lm).fit()
# Check the parameters obtained
lr.params
```

```
const    0.126894
area     0.462192
dtype: float64
```

```
plt.scatter(X_train_lm.iloc[:, 1], y_train)
plt.plot(X_train_lm.iloc[:, 1], 0.127 + 0.462*X_train_lm.iloc[:, 1],
'r')
plt.show()
```



```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:
0.283
Model:                            OLS   Adj. R-squared:
0.281
Method:                 Least Squares   F-statistic:
149.6
Date:                Thu, 17 Apr 2025   Prob (F-statistic):
3.15e-29
Time:                        14:26:03   Log-Likelihood:
227.23
No. Observations:                 381   AIC:
-450.5
Df Residuals:                     379   BIC:
-442.6
Df Model:                           1

Covariance Type:            nonrobust

==============================================================================
```

```
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const          0.1269      0.013      9.853      0.000       0.102
0.152
area           0.4622      0.038     12.232      0.000       0.388
0.536
==============================================================================
========
Omnibus:                       67.313   Durbin-Watson:
2.018
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
143.063
Skew:                           0.925   Prob(JB):
8.59e-32
Kurtosis:                       5.365   Cond. No.
5.99
==============================================================================
========

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

```
X_train_lm = X_train[['area', 'bathrooms']]

import statsmodels.api as sm
X_train_lm = sm.add_constant(X_train_lm)
lr = sm.OLS(y_train, X_train_lm).fit()
lr.params
```

```
const        0.104589
area         0.398396
bathrooms    0.298374
dtype: float64
```

```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:
0.480
Model:                            OLS   Adj. R-squared:
0.477
Method:                 Least Squares   F-statistic:
174.1
Date:                Thu, 17 Apr 2025   Prob (F-statistic):
```

```
bathrooms    0.259978
bedrooms     0.181863
dtype: float64
```

```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:
0.505
Model:                            OLS   Adj. R-squared:
0.501
Method:                 Least Squares   F-statistic:
128.2
Date:                Thu, 17 Apr 2025   Prob (F-statistic):
3.12e-57
Time:                        14:31:18   Log-Likelihood:
297.76
No. Observations:                 381   AIC:
-587.5
Df Residuals:                     377   BIC:
-571.7
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
const          0.0414      0.018      2.292      0.022       0.006
0.077
area           0.3922      0.032     12.279      0.000       0.329
0.455
bathrooms      0.2600      0.026     10.033      0.000       0.209
0.311
bedrooms       0.1819      0.041      4.396      0.000       0.101
0.263
==============================================================================
Omnibus:                       50.037   Durbin-Watson:
2.136
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
124.806
Skew:                           0.648   Prob(JB):
```

```
Method:                 Least Squares   F-statistic:
60.40
Date:                Thu, 17 Apr 2025   Prob (F-statistic):
8.83e-83
Time:                        14:42:03   Log-Likelihood:
381.79
No. Observations:                 381   AIC:
-735.6
Df Residuals:                     367   BIC:
-680.4
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                  coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
const             0.0200      0.021      0.955      0.340      -
0.021       0.061
area              0.2347      0.030      7.795      0.000
0.175       0.294
bedrooms          0.0467      0.037      1.267      0.206      -
0.026       0.119
bathrooms         0.1908      0.022      8.679      0.000
0.148       0.234
stories           0.1085      0.019      5.661      0.000
0.071       0.146
mainroad          0.0504      0.014      3.520      0.000
0.022       0.079
guestroom         0.0304      0.014      2.233      0.026
0.004       0.057
basement          0.0216      0.011      1.943      0.053      -
0.000       0.043
hotwaterheating   0.0849      0.022      3.934      0.000
0.042       0.127
airconditioning   0.0669      0.011      5.899      0.000
0.045       0.089
parking           0.0607      0.018      3.365      0.001
0.025       0.096
prefarea          0.0594      0.012      5.040      0.000
0.036       0.083
semi-furnished    0.0009      0.012      0.078      0.938      -
0.022       0.024
unfurnished      -0.0310      0.013     -2.440      0.015      -
0.056      -0.006
```

```
7.92e-28
Kurtosis:                       5.487   Cond. No.
8.87
==============================================================================
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

```
housing.columns
```

```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories',
'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'semi-furnished', 'unfurnished'],
      dtype='object')
```

```
import statsmodels.api as sm

X train lm = sm.add constant(X train)
lr_1 = sm.OLS(y_train, X_train_lm).fit()
lr_1.params
```

```
const              0.020033
area               0.234664
bedrooms           0.046735
bathrooms          0.190823
stories            0.108516
mainroad           0.050441
guestroom          0.030428
basement           0.021595
hotwaterheating    0.084863
airconditioning    0.066881
parking            0.060735
prefarea           0.059428
semi-furnished     0.000921
unfurnished       -0.031006
dtype: float64
```

```
print(lr_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:
0.681
Model:                            OLS   Adj. R-squared:
0.670
========
Omnibus:                       93.687   Durbin-Watson:
2.093
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
304.917
Skew:                           1.091   Prob(JB):
6.14e-67
Kurtosis:                       6.801   Cond. No.
14.6
==============================================================================
========
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()
vif['Features'] = X train.columns
vif['VIF'] = [variance inflation_factor(X_train.values, i) for i in
range(X train.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)

vif
```

	Features	VIF
1	bedrooms	7.33
4	mainroad	6.02
0	area	4.67
3	stories	2.70
11	semi-furnished	2.19
9	parking	2.12
6	basement	2.02
12	unfurnished	1.82
8	airconditioning	1.77
2	bathrooms	1.67
10	prefarea	1.51
5	guestroom	1.47
7	hotwaterheating	1.14

Conclusion:-

Here we see that the vif score of bedrooms and mainroad is grater than 5 so, the model is not best fitted so we rejected this .