1_Multiple Linear Regression

October 17, 2021

1 Multiple Linear Regression

1.1 Importing the libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
```

[3]:
```
from sklearn.model_selection import train_test_split
```

[4]:
```
from sklearn.linear_model import LinearRegression
```

[5]:
```
import statsmodels.api as sm
```

[6]:
```
import seaborn as sns
```

1.2 Importing the data set

[7]:
```
data_set = pd.read_csv("housing.csv")
print(data_set)
```

```
        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
..        …     …         …          …        …        …         …
540   1820000  3000         2          1        1      yes        no      yes
541   1767150  2400         3          1        1       no        no       no
542   1750000  3620         2          1        1      yes        no       no
543   1750000  2910         3          1        1       no        no       no
544   1750000  3850         3          1        2      yes        no       no
```

```
    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
..              …              …     …      …              …
540              no               no        2       no      unfurnished
541              no               no        0       no   semi-furnished
542              no               no        0       no      unfurnished
543              no               no        0       no        furnished
544              no               no        0       no      unfurnished

[545 rows x 13 columns]
```

[8]:
```
x = data_set.iloc[:,1:].values
y = data_set.iloc[:,0].values
```

[9]:
```
print(x)
```

```
[[7420 4 2 … 2 'yes' 'furnished']
 [8960 4 4 … 3 'no' 'furnished']
 [9960 3 2 … 2 'yes' 'semi-furnished']
 …
 [3620 2 1 … 0 'no' 'unfurnished']
 [2910 3 1 … 0 'no' 'furnished']
 [3850 3 1 … 0 'no' 'unfurnished']]
```

[10]:
```
print(y)
```

```
[13300000 12250000 12250000 12215000 11410000 10850000 10150000 10150000
  9870000  9800000  9800000  9681000  9310000  9240000  9240000  9100000
  9100000  8960000  8890000  8855000  8750000  8680000  8645000  8645000
  8575000  8540000  8463000  8400000  8400000  8400000  8400000  8400000
  8295000  8190000  8120000  8080940  8043000  7980000  7962500  7910000
  7875000  7840000  7700000  7700000  7560000  7560000  7525000  7490000
  7455000  7420000  7420000  7420000  7350000  7350000  7350000  7350000
  7343000  7245000  7210000  7210000  7140000  7070000  7070000  7035000
  7000000  6930000  6930000  6895000  6860000  6790000  6790000  6755000
  6720000  6685000  6650000  6650000  6650000  6650000  6650000  6650000
  6629000  6615000  6615000  6580000  6510000  6510000  6510000  6475000
  6475000  6440000  6440000  6419000  6405000  6300000  6300000  6300000
  6300000  6300000  6293000  6265000  6230000  6230000  6195000  6195000
  6195000  6160000  6160000  6125000  6107500  6090000  6090000  6090000
  6083000  6083000  6020000  6020000  6020000  5950000  5950000  5950000
  5950000  5950000  5950000  5950000  5950000  5943000  5880000  5880000
  5873000  5873000  5866000  5810000  5810000  5810000  5803000  5775000
  5740000  5740000  5740000  5740000  5740000  5652500  5600000  5600000
  5600000  5600000  5600000  5600000  5600000  5600000  5600000  5565000
```

5565000	5530000	5530000	5530000	5523000	5495000	5495000	5460000
5460000	5460000	5460000	5425000	5390000	5383000	5320000	5285000
5250000	5250000	5250000	5250000	5250000	5250000	5250000	5250000
5250000	5243000	5229000	5215000	5215000	5215000	5145000	5145000
5110000	5110000	5110000	5110000	5075000	5040000	5040000	5040000
5040000	5033000	5005000	4970000	4970000	4956000	4935000	4907000
4900000	4900000	4900000	4900000	4900000	4900000	4900000	4900000
4900000	4900000	4900000	4900000	4893000	4893000	4865000	4830000
4830000	4830000	4830000	4795000	4795000	4767000	4760000	4760000
4760000	4753000	4690000	4690000	4690000	4690000	4690000	4690000
4655000	4620000	4620000	4620000	4620000	4620000	4613000	4585000
4585000	4550000	4550000	4550000	4550000	4550000	4550000	4550000
4543000	4543000	4515000	4515000	4515000	4515000	4480000	4480000
4480000	4480000	4480000	4473000	4473000	4473000	4445000	4410000
4410000	4403000	4403000	4403000	4382000	4375000	4340000	4340000
4340000	4340000	4340000	4319000	4305000	4305000	4277000	4270000
4270000	4270000	4270000	4270000	4270000	4235000	4235000	4200000
4200000	4200000	4200000	4200000	4200000	4200000	4200000	4200000
4200000	4200000	4200000	4200000	4200000	4200000	4200000	4200000
4193000	4193000	4165000	4165000	4165000	4130000	4130000	4123000
4098500	4095000	4095000	4095000	4060000	4060000	4060000	4060000
4060000	4025000	4025000	4025000	4007500	4007500	3990000	3990000
3990000	3990000	3990000	3920000	3920000	3920000	3920000	3920000
3920000	3920000	3885000	3885000	3850000	3850000	3850000	3850000
3850000	3850000	3850000	3836000	3815000	3780000	3780000	3780000
3780000	3780000	3780000	3773000	3773000	3773000	3745000	3710000
3710000	3710000	3710000	3710000	3703000	3703000	3675000	3675000
3675000	3675000	3640000	3640000	3640000	3640000	3640000	3640000
3640000	3640000	3640000	3633000	3605000	3605000	3570000	3570000
3570000	3570000	3535000	3500000	3500000	3500000	3500000	3500000
3500000	3500000	3500000	3500000	3500000	3500000	3500000	3500000
3500000	3500000	3500000	3500000	3493000	3465000	3465000	3465000
3430000	3430000	3430000	3430000	3430000	3430000	3423000	3395000
3395000	3395000	3360000	3360000	3360000	3360000	3360000	3360000
3360000	3360000	3353000	3332000	3325000	3325000	3290000	3290000
3290000	3290000	3290000	3290000	3290000	3290000	3255000	3255000
3234000	3220000	3220000	3220000	3220000	3150000	3150000	3150000
3150000	3150000	3150000	3150000	3150000	3150000	3143000	3129000
3118850	3115000	3115000	3115000	3087000	3080000	3080000	3080000
3080000	3045000	3010000	3010000	3010000	3010000	3010000	3010000
3010000	3003000	2975000	2961000	2940000	2940000	2940000	2940000
2940000	2940000	2940000	2940000	2870000	2870000	2870000	2870000
2852500	2835000	2835000	2835000	2800000	2800000	2730000	2730000
2695000	2660000	2660000	2660000	2660000	2660000	2660000	2660000
2653000	2653000	2604000	2590000	2590000	2590000	2520000	2520000
2520000	2485000	2485000	2450000	2450000	2450000	2450000	2450000
2450000	2408000	2380000	2380000	2380000	2345000	2310000	2275000
2275000	2275000	2240000	2233000	2135000	2100000	2100000	2100000

```
 1960000  1890000  1890000  1855000  1820000  1767150  1750000  1750000
 1750000]
```

2 Taking care of missing data

[11]:
```python
imputer = SimpleImputer(missing_values = np.nan, strategy="mean")
imputer.fit(x[:,0:4])
x[:,0:4] = imputer.transform(x[:,0:4])
print(x)
```

```
[[7420.0 4.0 2.0 … 2 'yes' 'furnished']
 [8960.0 4.0 4.0 … 3 'no' 'furnished']
 [9960.0 3.0 2.0 … 2 'yes' 'semi-furnished']
 …
 [3620.0 2.0 1.0 … 0 'no' 'unfurnished']
 [2910.0 3.0 1.0 … 0 'no' 'furnished']
 [3850.0 3.0 1.0 … 0 'no' 'unfurnished']]
```

2.1 Encoding categorical data

[12]:
```python
le = LabelEncoder()
for i in range(4,9):
    x[:,i]=le.fit_transform(x[:, i])
x[:,10] = le.fit_transform(x[:, 10])
df2 = pd.DataFrame(x,
                  ␣
 ↪columns=['area','bedrooms','bathrooms','stories','mainroad','guestroom','basement','hotwate
print(df2)
```

```
       area bedrooms bathrooms stories mainroad guestroom basement  \
0    7420.0      4.0       2.0     3.0        1         0        0
1    8960.0      4.0       4.0     4.0        1         0        0
2    9960.0      3.0       2.0     2.0        1         0        1
3    7500.0      4.0       2.0     2.0        1         0        1
4    7420.0      4.0       1.0     2.0        1         1        1
..      …       …        …      …       …        …       …
540  3000.0      2.0       1.0     1.0        1         0        1
541  2400.0      3.0       1.0     1.0        0         0        0
542  3620.0      2.0       1.0     1.0        1         0        0
543  2910.0      3.0       1.0     1.0        0         0        0
544  3850.0      3.0       1.0     2.0        1         0        0

    hotwaterheating airconditioning parking prefarea furnishingstatus
0                 0               1       2        1        furnished
1                 0               1       3        0        furnished
2                 0               0       2        1   semi-furnished
3                 0               1       3        1        furnished
4                 0               1       2        0        furnished
```

```
..                   …                 …      …       …                …
540                   0                 0       2         0      unfurnished
541                   0                 0       0         0   semi-furnished
542                   0                 0       0         0      unfurnished
543                   0                 0       0         0        furnished
544                   0                 0       0         0      unfurnished

[545 rows x 12 columns]
```

[13]:
```
ct = ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[-1])],␣
 ↪remainder ='passthrough')
x = ct.fit_transform(x)
```

[14]:
```
print(x)
```

```
[[1.0 0.0 0.0 … 1 2 1]
 [1.0 0.0 0.0 … 1 3 0]
 [0.0 1.0 0.0 … 0 2 1]
 …
 [0.0 0.0 1.0 … 0 0 0]
 [1.0 0.0 0.0 … 0 0 0]
 [0.0 0.0 1.0 … 0 0 0]]
```

[15]:
```
x = x[:, 1:]
```

2.2 Splitting the dataset into the Training set and Test set

[16]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state␣
 ↪= 0)
```

2.3 Training Multiple linear Regression Model on Training set

[17]:
```
Regressor = LinearRegression()
Regressor.fit(x_train,y_train)
```

[17]: LinearRegression()

2.4 Predict Test Result

[18]:
```
y_predict = Regressor.predict(x_test)
print(y_predict)
```

```
[ 3950288.61876196  6173868.81883061  4483635.98836272  7258732.75105198
  2836727.58490499  7032947.0974906   3203851.47112406  3270994.00904074
  3472554.03645932  8289978.32623682  6605321.62954614  3723366.23684106
  3812376.95976104  4548966.84544599  4020476.3484966   1969836.22090165
  4057262.98087856  3704586.86711732  3282767.93188815  4609423.64909571
  5968243.73637128  6363698.62063799  4751300.32389003  2659595.27633066
  5305573.24662102  5680819.58784482  5404106.90027122  5543050.5219253
```

```
 5768360.4798223   5801753.70839269  3389277.96110617  6399092.02678457
 7081030.31411738  2913042.40387681  4498664.01335446  5210561.68059354
 5013457.84122302  3707596.71347599  2916603.45485358  3937761.75634098
 8041334.20180879  4942174.6114205   6432605.21981742  3511338.78156419
 3813475.39540813  6434856.19540018  4447687.02885148  2696243.71724915
 4180018.706258    6455973.25779215  4056226.34306799  7124571.30073128
 2530661.67791795  3033278.46419645  3500830.32062843  5119451.01676899
 7110973.93249708  4127705.79986428  2970005.36861369  4325732.61744607
 5986119.70766227  6824682.68794762  3325637.45729309  7191804.55935353
 2609468.55099856  5056521.66455864  6636269.77589372  2565659.89128227
 3751294.03758704  5080427.99370204  4281895.68812439  7361447.18275847
 5088033.19021914  6022539.93047597  4176648.2040903   4639478.54662236
 2898083.34640804  7564393.66040171  2583102.74086279  3764386.73199941
 4281895.68812439  6064669.4160839   5199726.50699991  5402615.00751849
 3900783.41794339  4206866.26507446  4785571.46071508  5125782.90220209
 3843109.12191378  4373515.96292211  3233779.57826289  5800152.85515771
 3086788.94075898  3736808.50597401  4475695.33317646 10490600.69498214
 3044861.09249697  7172608.23555207  4348859.16641385  4508307.36728241
 6607800.84902966  3393091.94230324  4545560.49433471  3313363.08812188
 7340959.28328457  5235408.60082988  4134159.03053657  5058911.23363261
 6279957.32173079]
```

[19]:
```
df = pd.DataFrame({"Acctual value":y_test,"predicted value":np.round(y_predict,␣
↪2),"difference":y_test-np.round(y_predict, 2)})
```

[20]:
```
print(df)
```

```
     Acctual value  predicted value  difference
0          4585000       3950288.62   634711.38
1          6083000       6173868.82   -90868.82
2          4007500       4483635.99  -476135.99
3          6930000       7258732.75  -328732.75
4          2940000       2836727.58   103272.42
..             …               …           …
104        6650000       7340959.28  -690959.28
105        5810000       5235408.60   574591.40
106        4123000       4134159.03   -11159.03
107        3080000       5058911.23 -1978911.23
108        5530000       6279957.32  -749957.32

[109 rows x 3 columns]
```

2.5 Building the optimal model using Backward Elimination

[21]:
```
x = np.append(arr = np.ones((545, 1)).astype(int), values = x, axis = 1)
print(x)
```

```
[[1 0.0 0.0 … 1 2 1]
 [1 0.0 0.0 … 1 3 0]
```

```
 [1 1.0 0.0 … 0 2 1]
 …
 [1 0.0 1.0 … 0 0 0]
 [1 0.0 0.0 … 0 0 0]
 [1 0.0 1.0 … 0 0 0]]
```

[22]:
```
x_opt = x[:, [0, 1, 2, 3, 4, 5]]
x_opt = x_opt.astype(np.float64)
print(x_opt)
```

```
[[1.00e+00 0.00e+00 0.00e+00 7.42e+03 4.00e+00 2.00e+00]
 [1.00e+00 0.00e+00 0.00e+00 8.96e+03 4.00e+00 4.00e+00]
 [1.00e+00 1.00e+00 0.00e+00 9.96e+03 3.00e+00 2.00e+00]
 …
 [1.00e+00 0.00e+00 1.00e+00 3.62e+03 2.00e+00 1.00e+00]
 [1.00e+00 0.00e+00 0.00e+00 2.91e+03 3.00e+00 1.00e+00]
 [1.00e+00 0.00e+00 1.00e+00 3.85e+03 3.00e+00 1.00e+00]]
```

[23]:
```
regressor_OLS = sm.OLS(endog = y, exog = x_opt).fit()
regressor_OLS.summary()
```

[23]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.514
Model:                            OLS   Adj. R-squared:                  0.509
Method:                 Least Squares   F-statistic:                     113.8
Date:                Sun, 17 Oct 2021   Prob (F-statistic):           5.59e-82
Time:                        12:40:16   Log-Likelihood:                -8447.2
No. Observations:                 545   AIC:                         1.691e+04
Df Residuals:                     539   BIC:                         1.693e+04
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       4.696e+05   2.92e+05      1.607      0.109   -1.04e+05    1.04e+06
x1         -2.808e+05   1.42e+05     -1.983      0.048   -5.59e+05   -2620.471
x2          -7.93e+05   1.52e+05     -5.234      0.000   -1.09e+06   -4.95e+05
x3           358.8673     26.769     13.406      0.000     306.283     411.451
x4          3.756e+05   8.26e+04      4.546      0.000    2.13e+05    5.38e+05
x5           1.33e+06   1.22e+05     10.869      0.000    1.09e+06    1.57e+06
==============================================================================
Omnibus:                       75.630   Durbin-Watson:                   1.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              152.634
Skew:                           0.789   Prob(JB):                     7.18e-34
Kurtosis:                       5.058   Cond. No.                     3.12e+04
```

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

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.12e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[24]:
```python
x_opt = x[:, [0, 1, 3, 4, 5]]
x_opt = x_opt.astype(np.float64)
regressor_OLS = sm.OLS(endog = y, exog = x_opt).fit()
regressor_OLS.summary()
```

[24]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.489
Model:                            OLS   Adj. R-squared:                  0.485
Method:                 Least Squares   F-statistic:                     129.1
Date:                Sun, 17 Oct 2021   Prob (F-statistic):           2.72e-77
Time:                        12:40:16   Log-Likelihood:                -8460.7
No. Observations:                 545   AIC:                         1.693e+04
Df Residuals:                     540   BIC:                         1.695e+04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -2.229e+05   2.67e+05     -0.835      0.404   -7.47e+05    3.01e+05
x1          1.589e+05   1.17e+05      1.361      0.174   -7.05e+04    3.88e+05
x2           378.8844     27.134     13.963      0.000     325.583     432.186
x3           4.02e+05   8.45e+04      4.759      0.000    2.36e+05    5.68e+05
x4          1.384e+06   1.25e+05     11.080      0.000    1.14e+06    1.63e+06
==============================================================================
Omnibus:                       74.733   Durbin-Watson:                   0.955
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              156.197
Skew:                           0.768   Prob(JB):                     1.21e-34
Kurtosis:                       5.126   Cond. No.                     2.67e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.67e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

"""

[25]:
```
x_opt = x[:, [0, 3, 4, 5]]
x_opt = x_opt.astype(np.float64)
regressor_OLS = sm.OLS(endog = y, exog = x_opt).fit()
regressor_OLS.summary()
```

[25]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.487
Model:                            OLS   Adj. R-squared:                  0.484
Method:                 Least Squares   F-statistic:                     171.3
Date:                Sun, 17 Oct 2021   Prob (F-statistic):           4.80e-78
Time:                        12:40:16   Log-Likelihood:                -8461.6
No. Observations:                 545   AIC:                         1.693e+04
Df Residuals:                     541   BIC:                         1.695e+04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -1.732e+05   2.65e+05     -0.655      0.513   -6.93e+05    3.47e+05
x1           378.7628     27.155     13.948      0.000     325.420     432.105
x2          4.068e+05   8.45e+04      4.817      0.000    2.41e+05    5.73e+05
x3          1.386e+06   1.25e+05     11.089      0.000    1.14e+06    1.63e+06
==============================================================================
Omnibus:                       70.408   Durbin-Watson:                   0.952
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              142.930
Skew:                           0.738   Prob(JB):                     9.19e-32
Kurtosis:                       5.029   Cond. No.                     2.64e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.64e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[26]:
```
x_opt = x[:, [0, 3, 5]]
x_opt = x_opt.astype(np.float64)
regressor_OLS = sm.OLS(endog = y, exog = x_opt).fit()
regressor_OLS.summary()
```

[26]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.465
Model:                            OLS   Adj. R-squared:                  0.463
Method:                 Least Squares   F-statistic:                     235.6
Date:                Sun, 17 Oct 2021   Prob (F-statistic):           2.32e-74
Time:                        12:40:16   Log-Likelihood:                -8473.0
No. Observations:                 545   AIC:                         1.695e+04
Df Residuals:                     542   BIC:                         1.697e+04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       6.992e+05   1.97e+05      3.554      0.000    3.13e+05    1.09e+06
x1           390.1749     27.600     14.137      0.000     335.958     444.391
x2             1.6e+06   1.19e+05     13.422      0.000    1.37e+06    1.83e+06
==============================================================================
Omnibus:                       84.006   Durbin-Watson:                   0.922
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              182.089
Skew:                           0.840   Prob(JB):                     2.88e-40
Kurtosis:                       5.280   Cond. No.                     2.04e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.04e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[27]:
```python
x_opt = x[:, [0, 3]]
x_opt = x_opt.astype(np.float64)
regressor_OLS = sm.OLS(endog = y, exog = x_opt).fit()
regressor_OLS.summary()
```

[27]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.287
Model:                            OLS   Adj. R-squared:                  0.286
Method:                 Least Squares   F-statistic:                     218.9
Date:                Sun, 17 Oct 2021   Prob (F-statistic):           7.39e-42
Time:                        12:40:16   Log-Likelihood:                -8551.2
```

```
No. Observations:                 545   AIC:                         1.711e+04
Df Residuals:                     543   BIC:                         1.712e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.387e+06   1.74e+05     13.681      0.000    2.04e+06    2.73e+06
x1           461.9749     31.226     14.795      0.000     400.637     523.313
==============================================================================
Omnibus:                       92.668   Durbin-Watson:                   0.565
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              183.673
Skew:                           0.954   Prob(JB):                     1.31e-40
Kurtosis:                       5.108   Cond. No.                     1.44e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.44e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[28]:

```
print(x_opt)
```

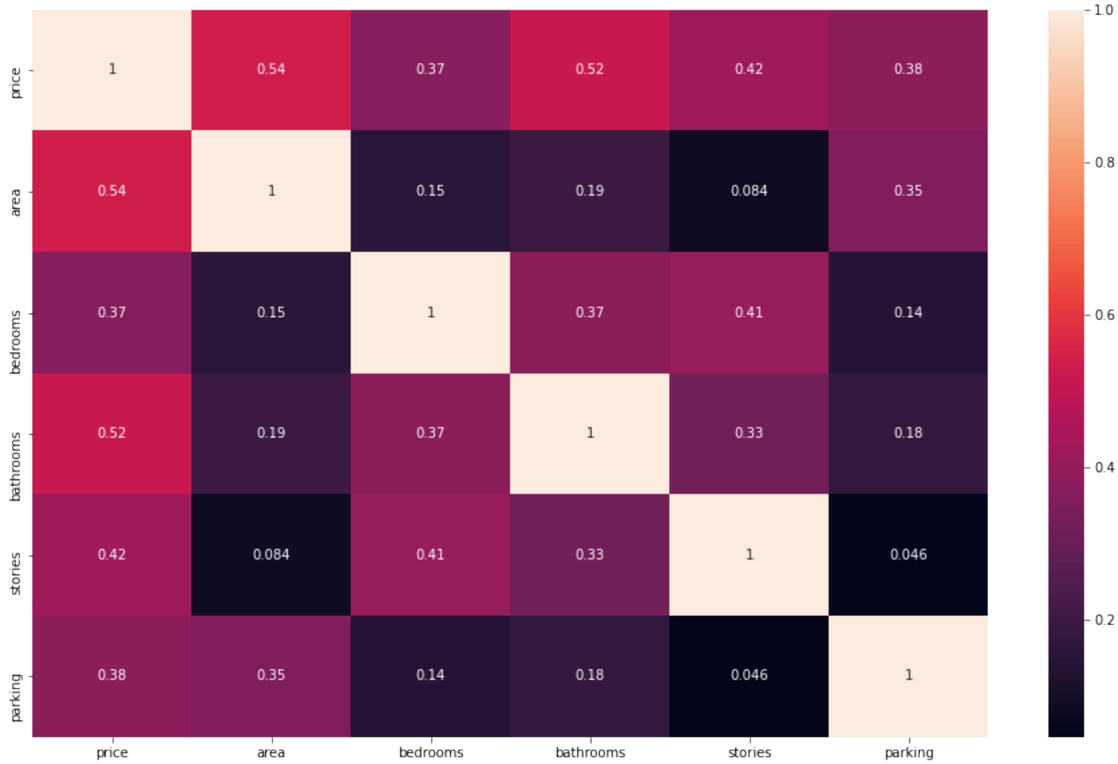
```
[[1.00e+00 7.42e+03]
 [1.00e+00 8.96e+03]
 [1.00e+00 9.96e+03]
 …
 [1.00e+00 3.62e+03]
 [1.00e+00 2.91e+03]
 [1.00e+00 3.85e+03]]
```

[29]:

```
%matplotlib inline
```

[30]:

```
plt.figure(figsize = (16,10))
sns.heatmap(data_set.corr(),annot= True)
plt.show()
```


price
area
bedrooms
bathrooms
stories
parking
1 0.54 0.37 0.52 0.42 0.38
0.54 1 0.15 0.19 0.084 0.35
0.37 0.15 1 0.37 0.41 0.14
0.52 0.19 0.37 1 0.33 0.18
0.42 0.084 0.41 0.33 1 0.046
0.38 0.35 0.14 0.18 0.046 1
1.0
0.8
0.6
0.4
0.2

```
[ ]:
```

```
[ ]:
```