vlkqbgccr

February 25, 2025

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

[2]:
```
df=pd.read_csv("Housing1.csv")
```

[3]:
```
df
```

[3]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished
..	…	…	…	…	…	…	…	…	…	…	…	…	…
540	1820000	3000	2	1	1	yes	no	yes	no	no	2	no	unfurnished
541	1767150	2400	3	1	1	no	no	no	no	no	0	no	semi-furnished
542	1750000	3620	2	1	1	yes	no	no	no	no	0	no	unfurnished
543	1750000	2910	3	1	1	no	no	no	no	no	0	no	furnished
544	1750000	3850	3	1	2	yes	no	no	no	no	0	no	unfurnished

[545 rows x 13 columns]

[4]:
```
df.isnull().sum()
```

[4]:
```
price              0
area               0
bedrooms           0
bathrooms          0
stories            0
mainroad           0
guestroom          0
basement           0
hotwaterheating    0
airconditioning    0
parking            0
prefarea           0
furnishingstatus   0
dtype: int64
```

[6]:
```
x = df.drop(['price'], axis = 1)
y = df['price']
```

[8]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.
 ↪2,random_state = 0)
```

[10]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['mainroad'] = le.fit_transform(df['mainroad'])
newdf=df
```

[11]:
```
df
```

[11]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	\
0	13300000	7420	4	2	3	1	no	
1	12250000	8960	4	4	4	1	no	
2	12250000	9960	3	2	2	1	no	
3	12215000	7500	4	2	2	1	no	
4	11410000	7420	4	1	2	1	yes	
..	…	…	…	…	…	…	…	
540	1820000	3000	2	1	1	1	no	
541	1767150	2400	3	1	1	0	no	
542	1750000	3620	2	1	1	1	no	
543	1750000	2910	3	1	1	0	no	
544	1750000	3850	3	1	2	1	no	

	basement	hotwaterheating	airconditioning	parking	prefarea	\
0	no	no	yes	2	yes	
1	no	no	yes	3	no	

```
2        yes              no              no        2      yes
3        yes              no             yes        3      yes
4        yes              no             yes        2       no
..       …                …               …       …      …
540      yes              no              no        2       no
541       no              no              no        0       no
542       no              no              no        0       no
543       no              no              no        0       no
544       no              no              no        0       no

    furnishingstatus
0          furnished
1          furnished
2     semi-furnished
3          furnished
4          furnished
..               …
540      unfurnished
541   semi-furnished
542      unfurnished
543        furnished
544      unfurnished

[545 rows x 13 columns]
```

[13]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['guestroom'] = le.fit_transform(df['guestroom'])
newdf=df
```

[14]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['basement'] = le.fit_transform(df['basement'])
newdf=df
```

[15]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['hotwaterheating'] = le.fit_transform(df['hotwaterheating'])
newdf=df
```

[16]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['airconditioning'] = le.fit_transform(df['airconditioning'])
newdf=df
```

[17]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
df['prefarea'] = le.fit_transform(df['prefarea'])
newdf=df
```

[18]:
```
df
```

[18]:
```
        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  prefarea  \
0           0                0                1        2         1
1           0                0                1        3         0
2           1                0                0        2         1
3           1                0                1        3         1
4           1                0                1        2         0
..        ...              ...              ...      ...       ...
540         1                0                0        2         0
541         0                0                0        0         0
542         0                0                0        0         0
543         0                0                0        0         0
544         0                0                0        0         0

    furnishingstatus
0          furnished
1          furnished
2     semi-furnished
3          furnished
4          furnished
..               ...
540      unfurnished
541   semi-furnished
542      unfurnished
543        furnished
544      unfurnished

[545 rows x 13 columns]
```

[20]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['furnishingstatus'] = le.fit_transform(df['furnishingstatus'])
newdf=df
```

[21]:
```
df
```

[21]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	0
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	0
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	1
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	0
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	0
..	…	…	…	…	…	…	…	…	…	…	…	…	…
540	1820000	3000	2	1	1	1	0	1	0	0	2	0	2
541	1767150	2400	3	1	1	0	0	0	0	0	0	0	0
542	1750000	3620	2	1	1	1	0	0	0	0	0	0	1
543	1750000	2910	3	1	1	0	0	0	0	0	0	0	0
544	1750000	3850	3	1	2	1	0	0	0	0	0	0	0

[545 rows x 13 columns]

[24]:
```
df.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    int32
 6   guestroom         545 non-null    int32
 7   basement          545 non-null    int32
 8   hotwaterheating   545 non-null    int32
 9   airconditioning   545 non-null    int32
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    int32
 12  furnishingstatus  545 non-null    int32
dtypes: int32(7), int64(6)
memory usage: 40.6 KB
```

[25]:
```
df.describe()
```

[25]:

	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

	mainroad	guestroom	basement	hotwaterheating	airconditioning
count	545.000000	545.000000	545.000000	545.000000	545.000000
mean	0.858716	0.177982	0.350459	0.045872	0.315596
std	0.348635	0.382849	0.477552	0.209399	0.465180
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000	0.000000
75%	1.000000	0.000000	1.000000	0.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

	parking	prefarea	furnishingstatus
count	545.000000	545.000000	545.000000

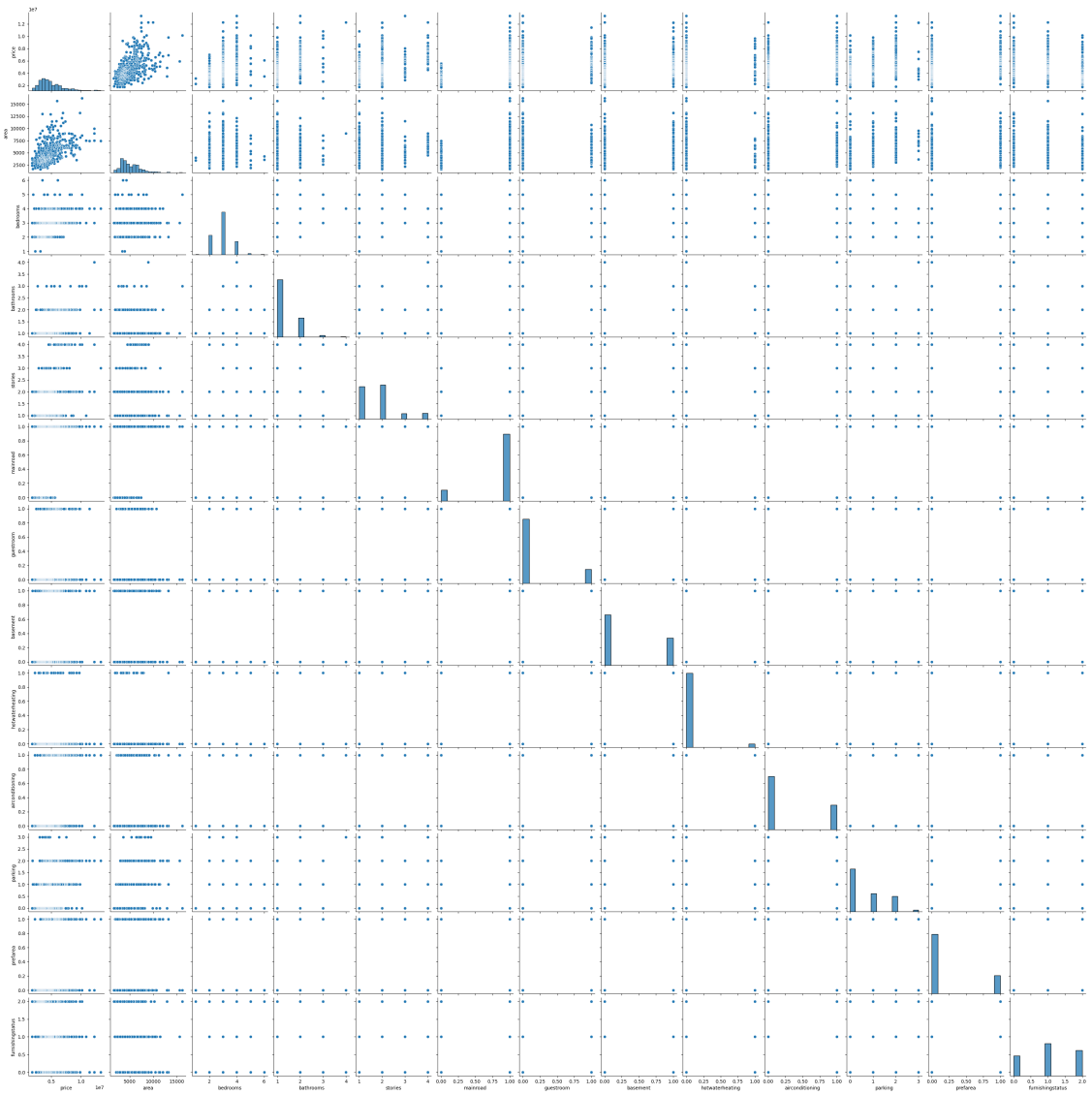
mean	0.693578	0.234862	1.069725
std	0.861586	0.424302	0.761373
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	1.000000
75%	1.000000	0.000000	2.000000
max	3.000000	1.000000	2.000000

[26]:
```
sns.pairplot(df)
```

```
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
```

```
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

```
[26]: <seaborn.axisgrid.PairGrid at 0x2ea458c64d0>
```



[27]:
```
df['price']
```

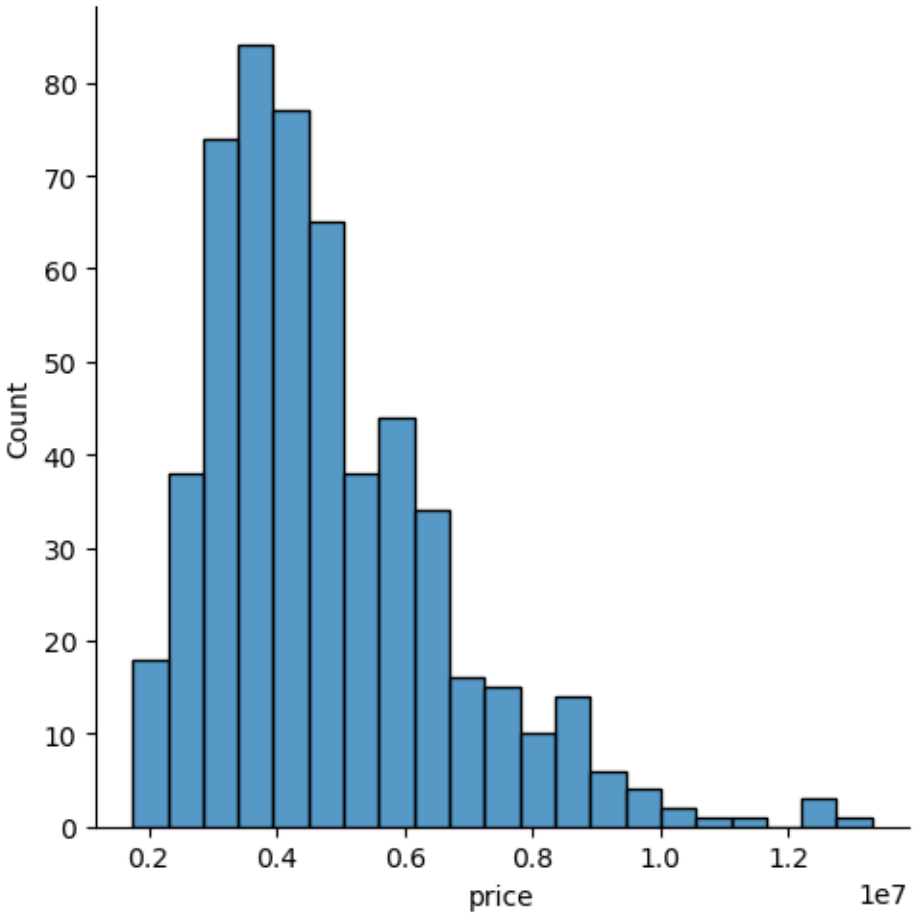
[27]:
```
0      13300000
1      12250000
2      12250000
3      12215000
4      11410000
         ...
540     1820000
541     1767150
542     1750000
543     1750000
544     1750000
```

```
Name: price, Length: 545, dtype: int64
```

[28]:
```
sns.displot(df['price'])
```

```
C:\Users\Vishwajeet Kulkarni\anaconda3\Lib\site-
packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

[28]:
```
<seaborn.axisgrid.FacetGrid at 0x2ea5102e450>
```



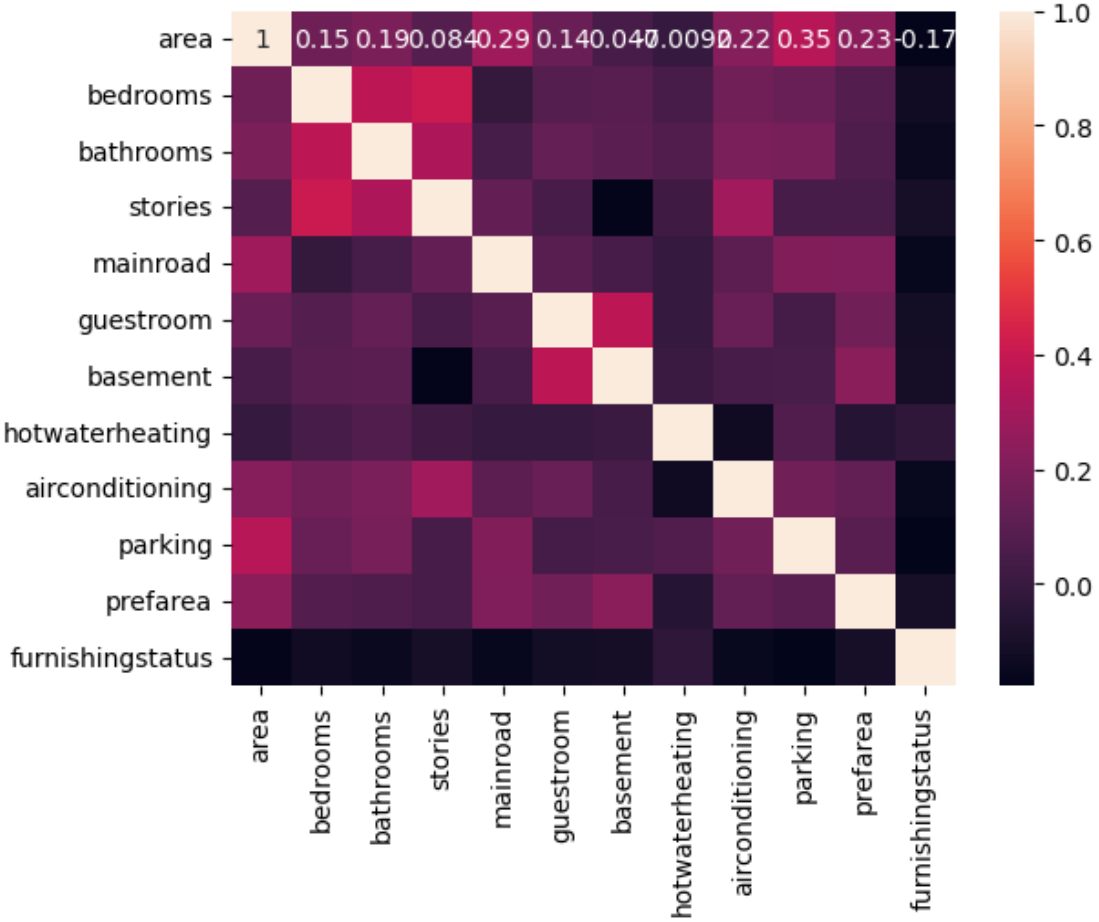
[29]:
```
df.columns
```

[29]:
```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'furnishingstatus'],
      dtype='object')
```

[32]:
```
x = df[['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
        'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
        'parking', 'prefarea', 'furnishingstatus']]
y = df['price']
```

[33]:
```
sns.heatmap(x.corr(), annot=True)
```

[33]: <Axes: >



[35]:
```
from sklearn.model_selection import train_test_split
```

[38]:
```
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=10)
```

[39]:
```
from sklearn.linear_model import LinearRegression
```

[40]:
```
lm = LinearRegression()
lm.fit(X_train, Y_train)
```

[40]:
```
LinearRegression()
```

[42]:
```
Y_pred = lm.predict(X_test)
print("Predictions:", Y_pred)
```

```
Predictions: [ 2700404.37444962  3696574.77906131  6107673.11773901
3724468.65864684
  6557661.20422199  6930111.52196603  3514414.46487381  3134354.87200507
  5694076.14703775  3295867.57344321  5124605.43704662  2224896.88633438
  4267817.74148458  4604261.70602523  6771346.76359304  2798678.43525882
  7446757.31282124  6265201.03289684  5212092.94398296  4193658.45167661
  2597604.1584731   2061583.01111558  5667557.40318702  5512825.79947958
  6723574.74457072  6083751.34833798  3668115.34968383  3653929.66549101
  3592236.00121165  3796916.6020339   4529767.85837463  5083192.53558171
  5550083.36658613  5468788.36577786  7670159.91803612  5102115.32457644
  6728166.04503339  2479610.80165039  5213910.43838254  5683238.05040679
  5089768.89892276  5058246.26649298  2905292.56101868  6595905.08167087
  5733630.96229429  7927640.36090784  8135626.52884708  6127483.66479094
  3582225.88370216  5002175.8547468   3273115.12115403  6176243.38577547
  3925550.99993589  4926680.38238415  2849895.93812145  5693064.328588
  5467981.16039256  3512037.59000936  3753761.51055862  4766885.31716492
  4693679.81692531  4827646.866127    5244970.08760391  3722031.46760491
  4087431.8920726   6876837.98661188  3157546.07506011  2554032.4739109
  2496236.92557867  7142847.94738654  4260931.6474929   6356021.68025515
  6123014.30769451  3677897.87058484  2885079.7332281   3477793.4680578
  3389239.81701097  4964750.67069313  9786750.95210684  4835958.37390079
  7312401.23730613  5251288.50369311  3819189.52825802  2900478.6266073
  4640055.49911233  2670310.05533309  4839075.16386885 10676455.12927325
  5769471.50033742  3135899.77595036  5569253.65719456  3339989.85950933
  6402364.3689004   6108646.44998769  3992344.17468643  3517237.96713829
  3336304.2712518   6648029.03081334  4896303.41150151  4915849.42390665
  5519175.92052042  4165827.28517975  4377037.97737514  4366360.08136088
  6940668.64144853  4181563.00942269  4000822.79800416  3297756.75957662
  5908495.92539094  2855178.15051111  4370125.86311565  5144114.09580452
  3807713.9118241   2823512.59255625  2909210.47361788  5357476.15744724
  3311533.67882539  2265797.89585087  5067594.95525651  4826960.89433245
  7635218.4539337   4300417.88946938  4710393.42826515  4080195.77652826
  7488144.45576757  3355813.59610503  4963208.94079417  4639597.25696116
  5320926.07479037  4619650.19881016  4968996.7593027   5359790.21486715
  6670872.41010281  4473536.42218945  3374763.19710612  4849509.60085026
  4701771.64510876  3066824.15260646  6963969.12465449  5013463.74775451
  5397294.93296288  8737012.9522541   6338222.342816    3483384.40450796
  3803751.68127549  6852011.4918532   6701913.95285577  4244519.74914909
  5959662.72963786  4438992.20746348  3906861.297176    3311679.99908496
```

```
 3940272.88142562  6408902.73835126  7924016.96949363  3022544.23532682
 3683089.2630826   4367156.30042295  3048013.80418355  8139479.17160768
 6683549.05230061  4192535.20991326  5696482.57559047  5639544.76188649
 6652893.85111408  8419618.58676101  4158141.49652541  4247034.79374007
 4814178.8812911   6684258.27979311  5240205.67901918  3707065.61870947
 4325857.17076026  6332267.73411369  5702334.17711302  3280667.17347986
 2747610.19873846  4169181.49981332  3400063.39831079  5417919.08983175
 4042118.50036779  6866875.92898863  4473792.93779819  8517932.54957861
 6831463.14074392  3385199.59423149  7196685.38071541  6320598.89304315
 3664489.80594782  6534689.30081931  2635898.44645544  5935413.62486946
 4851714.6808392   3349055.78057873  6772208.28941836  4891796.04882885
 5920142.33316128  5056166.92582672  2666083.35580623  5285566.29729186
 3387997.98510105  3549186.77128405  4348965.01989714  7365454.24750964
 2788273.29708192  5086324.54754065  3649427.70159077  2689960.9706658
 4095680.58795986  6437392.46220683  6024845.20591643  4413650.55727109
 4937984.4261025   7631539.0007662   4487739.63225614  2322760.94177663
 4735379.63564623  7506912.84544311]
```

[43]:
```
import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
model=lm.fit(X_train, Y_train)
```

[44]:
```
Ytrain_pred = lm.predict(X_train)
Ytest_pred = lm.predict(X_test)
```

[45]:
```
df=pd.DataFrame(Ytrain_pred,Y_train)
df=pd.DataFrame(Ytest_pred,Y_test)
```

[46]:
```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(Y_test, Ytest_pred)
print(mse)
mse = mean_squared_error(Ytrain_pred,Y_train)
print(mse)
```

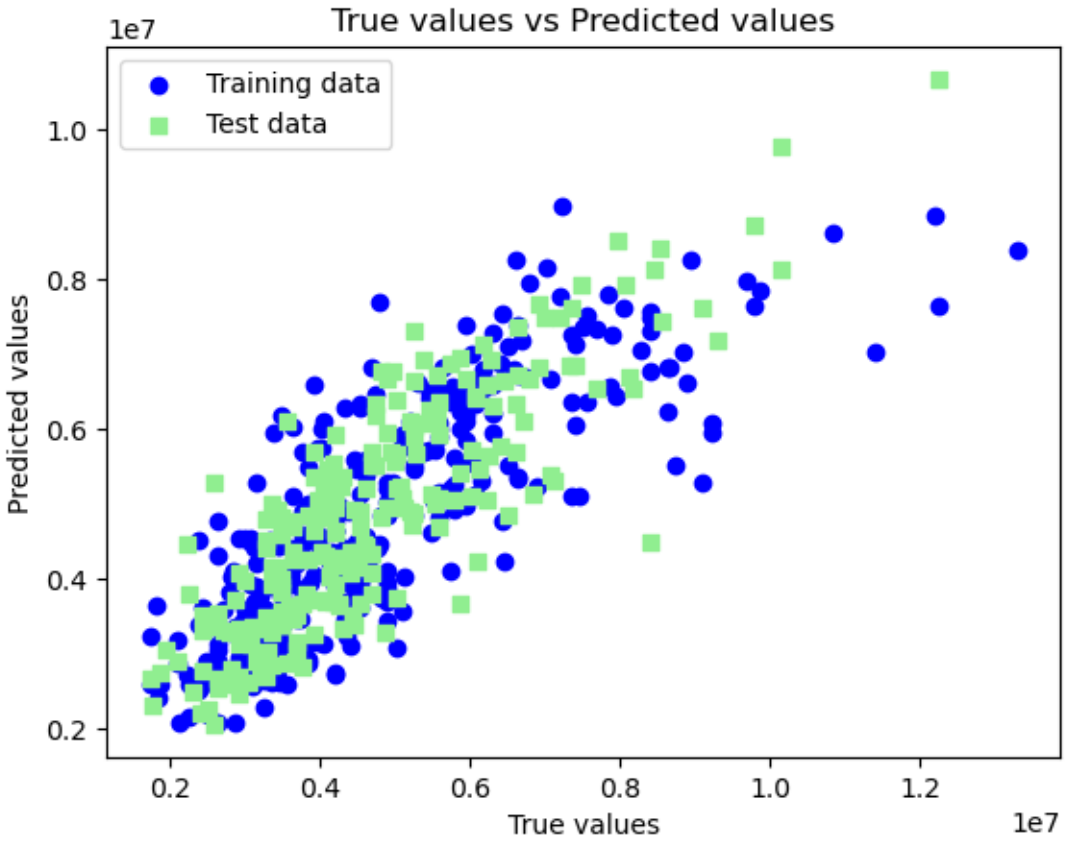
```
899748141018.0028
1302843420083.3718
```

[47]:
```
mse = mean_squared_error(Y_test, Ytest_pred)
print(mse)
```

```
899748141018.0028
```

[48]:
```
plt.scatter(Y_train, Ytrain_pred, c='blue', marker='o', label='Training data')
plt.scatter(Y_test, Ytest_pred, c='lightgreen', marker='s', label='Test data')
plt.xlabel('True values')
plt.ylabel('Predicted values')
plt.title("True values vs Predicted values")
```

```
plt.legend(loc='upper left')
plt.show()
```



[ ]: