Reading and Understanding the Data

In [1]:

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

In [2]:

```
import numpy as np
import pandas as pd
```

In [3]:

```
housing=pd.read_csv("datascience/HousingMLR.csv")
```

In [4]:

```
# Check the head of the dataset
housing.head()
```

Out[4]:

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

In [5]:

```
housing.shape
```

Out[5]:

```
(545, 13)
```

In [6]:

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
price               545 non-null int64
area                545 non-null int64
bedrooms            545 non-null int64
bathrooms           545 non-null int64
stories             545 non-null int64
mainroad            545 non-null object
guestroom           545 non-null object
basement            545 non-null object
hotwaterheating     545 non-null object
airconditioning     545 non-null object
parking             545 non-null int64
prefarea            545 non-null object
furnishingstatus    545 non-null object
dtypes: int64(6), object(7)
memory usage: 55.4+ KB
```

In [8]:

```
housing.describe()
```

Out[8]:

	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

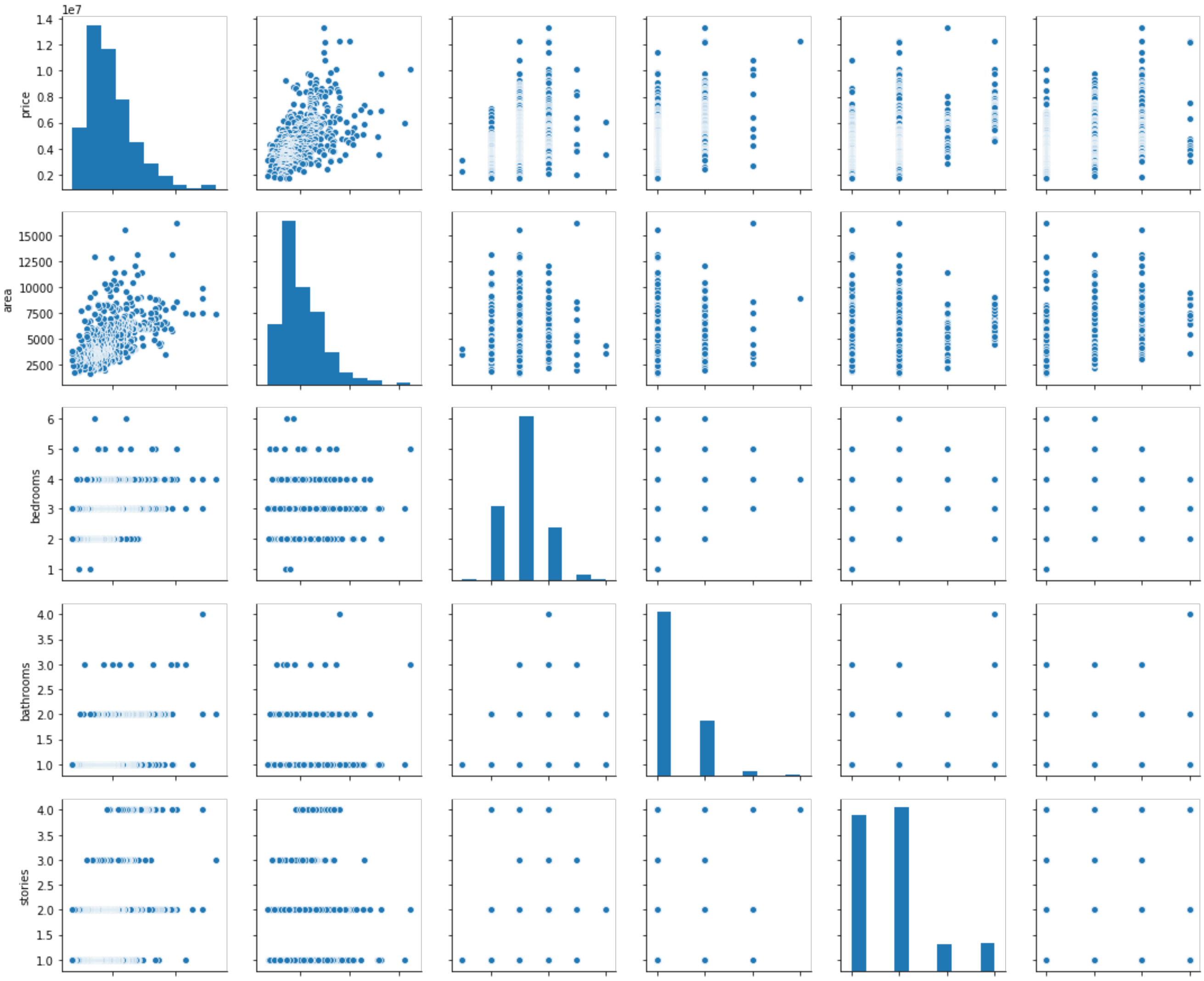
Visualising the Data

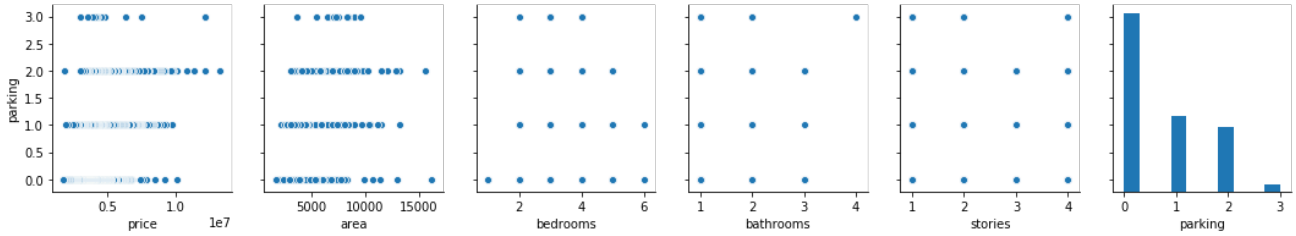
In [9]:

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

In [10]:

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

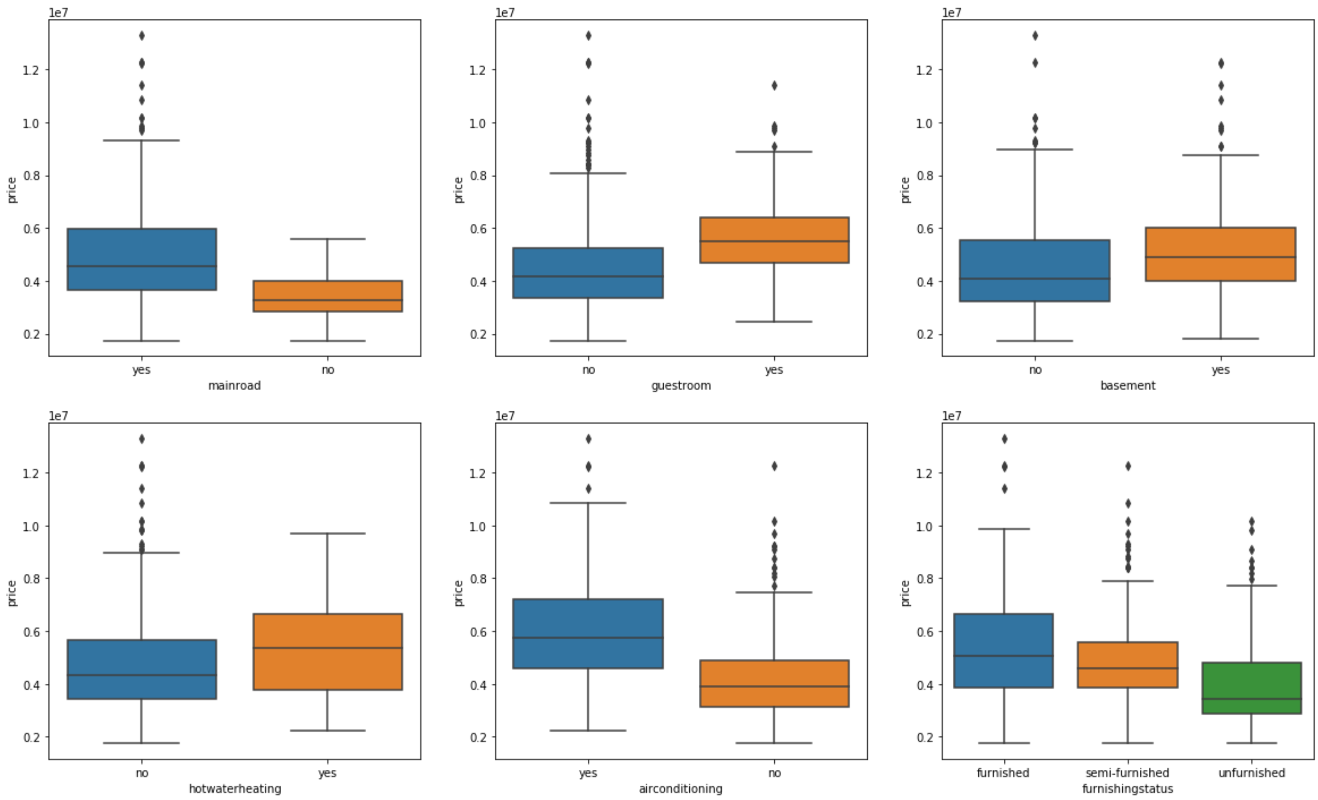




vizualising categorical variable

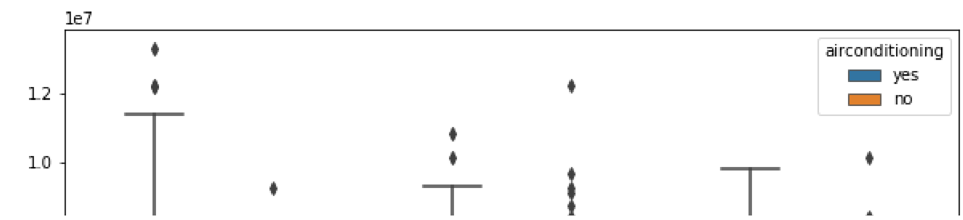
In [11]:

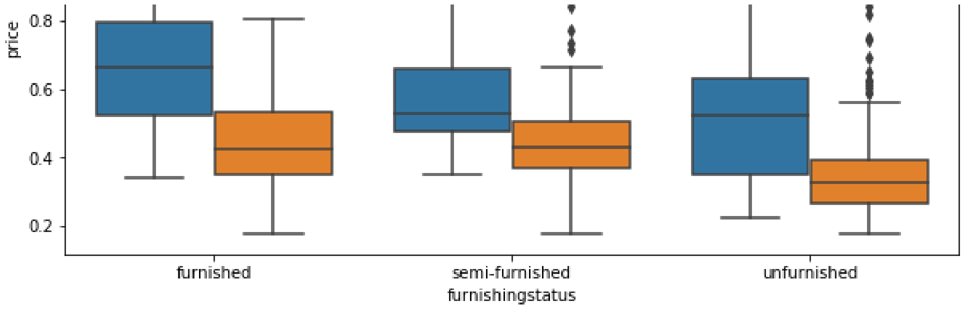
```
plt.figure(figsize=(20, 12))
plt.subplot(2,3,1)
sns.boxplot(x = 'mainroad', y = 'price', data = housing)
plt.subplot(2,3,2)
sns.boxplot(x = 'guestroom', y = 'price', data = housing)
plt.subplot(2,3,3)
sns.boxplot(x = 'basement', y = 'price', data = housing)
plt.subplot(2,3,4)
sns.boxplot(x = 'hotwaterheating',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



In [12]:

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





Data Preparation

In [13]:

```
# List of variables to map

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)
```

In [14]:

```
# Check the housing dataframe now

housing.head()
```

Out[14]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefare
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

dummy varaiables

In [15]:

```
# Get the dummy variables for the feature 'furnishingstatus' and store it in a new variable - 'sta
tus'
status = pd.get_dummies(housing['furnishingstatus'])
```

In [16]:

```
status.head()
```

Out[16]:

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

In [17]:

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

In [18]:

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

In [19]:

```python
housing.head()
```

Out[19]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefare
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

In [20]:

```python
# Drop 'furnishingstatus' as we have created the dummies for it

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

In [21]:

```python
housing.head()
```

Out[21]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefare
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

training and testing part

In [25]:

```python
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
)
```

In [26]:

```python
from sklearn.preprocessing import MinMaxScaler
```

In [27]:

```python
scaler = MinMaxScaler()
```

```
scaler = MinMaxScaler()
```

In [30]:

```
# Apply scaler() to all the columns except the 'yes-no' and 'dummy' variables
num_vars = ['area', 'bedrooms', 'bathrooms', 'stories', 'parking','price']

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

In [29]:

```
df_train.head()
```

Out[29]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking
359	0.169697	0.155227	0.4	0.0	0.000000	1	0	0	0	0	0.333333
19	0.615152	0.403379	0.4	0.5	0.333333	1	0	0	0	1	0.333333
159	0.321212	0.115628	0.4	0.5	0.000000	1	1	1	0	1	0.000000
35	0.548133	0.454417	0.4	0.5	1.000000	1	0	0	0	1	0.666667
28	0.575758	0.538015	0.8	0.5	0.333333	1	0	1	1	0	0.666667

In [31]:

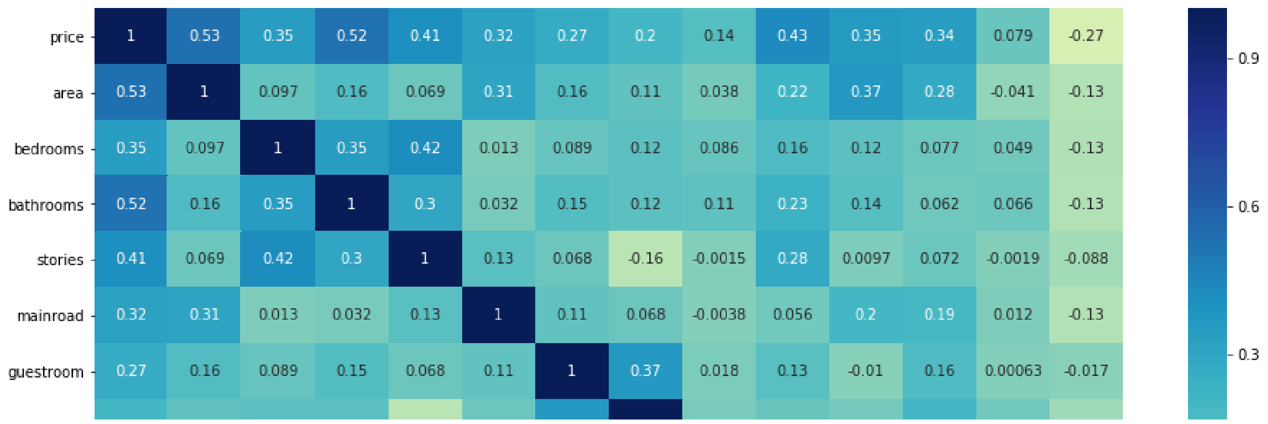
```
df_train.describe()
```

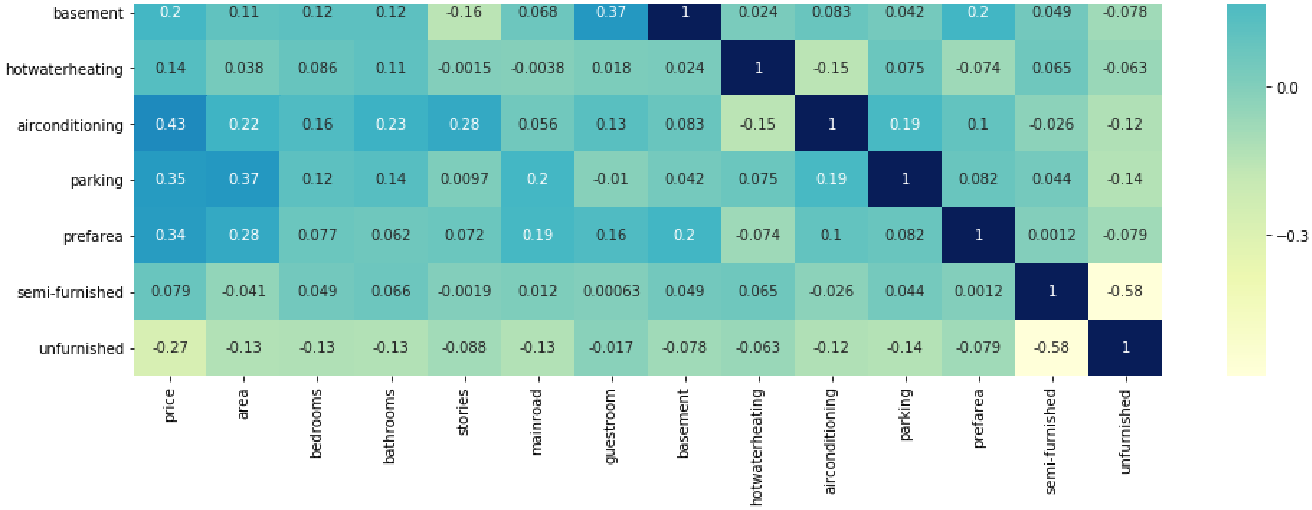
Out[31]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditionin
count	381.000000	381.000000	381.000000	381.000000	381.000000	381.000000	381.000000	381.000000	381.000000	381.00000
mean	0.260333	0.288710	0.386352	0.136483	0.268591	0.855643	0.170604	0.351706	0.052493	0.2992
std	0.157607	0.181420	0.147336	0.237325	0.295001	0.351913	0.376657	0.478131	0.223313	0.4585
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.151515	0.155227	0.200000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.0000
50%	0.221212	0.234424	0.400000	0.000000	0.333333	1.000000	0.000000	0.000000	0.000000	0.0000
75%	0.345455	0.398099	0.400000	0.500000	0.333333	1.000000	0.000000	1.000000	0.000000	1.0000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000

In [32]:

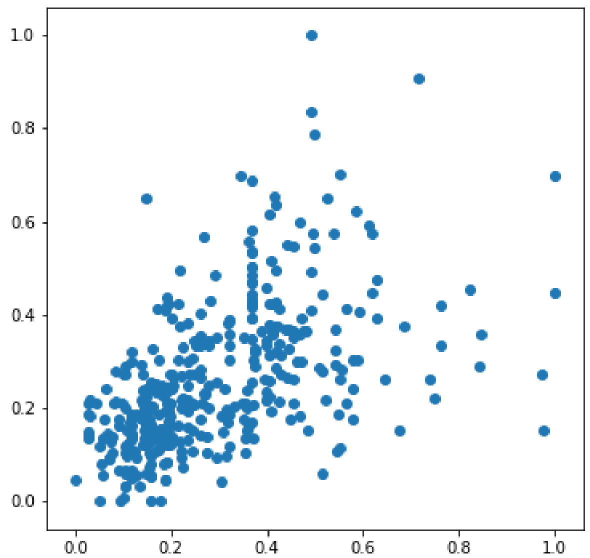
```
# Let's check the correlation coefficients to see which variables are highly correlated

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





In [33]:

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



In [34]:

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

building a linear modal

In [35]:

```
import statsmodels.api as sm

# Add a constant
X_train_lm = sm.add_constant(X_train[['area']])

# Create a first fitted model
lr = sm.OLS(y_train, X_train_lm).fit()
```

In [36]:

```
lr.params
```

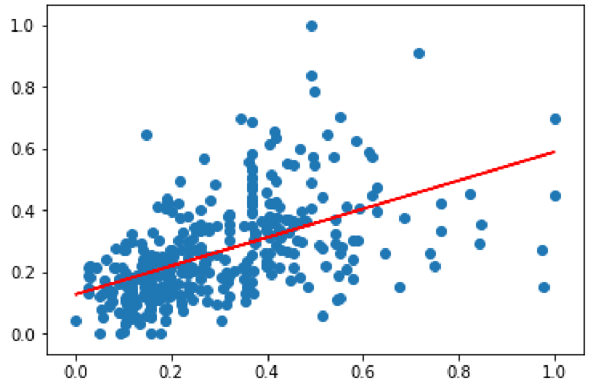
Out[36]:

```
const      0.126894
```

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

In [37]:

```
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()
```



In [38]:

```
# Print a summary of the linear regression model obtained
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:                Wed, 20 May 2020   Prob (F-statistic):           3.15e-29
Time:                        14:52:53   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
==============================================================================

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

In [39]:

```
# Assign all the feature variables to X
X_train_lm = X_train[['area', 'bathrooms']]
```

In [40]:

```
# Build a linear model

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
```

Out[40]:

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

In [41]:

```
# Check the summary
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:                Wed, 20 May 2020   Prob (F-statistic):           2.51e-54
Time:                        14:53:30   Log-Likelihood:                 288.24
No. Observations:                 381   AIC:                            -570.5
Df Residuals:                     378   BIC:                            -558.6
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1046      0.011      9.384      0.000       0.083       0.127
area           0.3984      0.033     12.192      0.000       0.334       0.463
bathrooms      0.2984      0.025     11.945      0.000       0.249       0.347
==============================================================================
Omnibus:                       62.839   Durbin-Watson:                   2.157
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              168.790
Skew:                           0.784   Prob(JB):                     2.23e-37
Kurtosis:                       5.859   Cond. No.                         6.17
==============================================================================

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

In [42]:

```
# Assign all the feature variables to X
X_train_lm = X_train[['area', 'bathrooms','bedrooms']]
```

In [43]:

```
# Build a linear model

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
```

Out[43]:

```
const        0.041352
area         0.392211
bathrooms    0.259978
bedrooms     0.181863
dtype: float64
```

In [44]:

```
# Print the summary of the model

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:                Wed, 20 May 2020   Prob (F-statistic):           3.12e-57
Time:                        14:54:16   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):                     7.92e-28
Kurtosis:                       5.487   Cond. No.                         8.87
==============================================================================

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

In [45]:

```python
# Check all the columns of the dataframe

housing.columns
```

Out[45]:

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

In [46]:

```python
#Build a linear model

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
```

Out[46]:

```
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
```

In [47]:

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

```
                            OLS Regression Results
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.681
Model:                            OLS   Adj. R-squared:                  0.670
Method:                 Least Squares   F-statistic:                     60.40
Date:                Wed, 20 May 2020   Prob (F-statistic):           8.83e-83
Time:                        14:54:54   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
==============================================================================
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
==============================================================================

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

checking vif

In [48]:

```python
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [49]:

```python
# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
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
```

Out[49]:

	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

	Features	VIF
2	bathrooms	1.67
10	prefarea	1.51
5	guestroom	1.47
7	hotwaterheating	1.14

In [50]:

```
X = X_train.drop('semi-furnished', 1,)
```

In [51]:

```
# Build a third fitted model
X_train_lm = sm.add_constant(X)

lr_2 = sm.OLS(y_train, X_train_lm).fit()
```

In [52]:

```
print(lr_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.681
Model:                            OLS   Adj. R-squared:                  0.671
Method:                 Least Squares   F-statistic:                     65.61
Date:                Wed, 20 May 2020   Prob (F-statistic):           1.07e-83
Time:                        14:56:20   Log-Likelihood:                 381.79
No. Observations:                 381   AIC:                            -737.6
Df Residuals:                     368   BIC:                            -686.3
Df Model:                          12
Covariance Type:            nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
const               0.0207      0.019      1.098      0.273      -0.016       0.058
area                0.2344      0.030      7.845      0.000       0.176       0.293
bedrooms            0.0467      0.037      1.268      0.206      -0.026       0.119
bathrooms           0.1909      0.022      8.697      0.000       0.148       0.234
stories             0.1085      0.019      5.669      0.000       0.071       0.146
mainroad            0.0504      0.014      3.524      0.000       0.022       0.079
guestroom           0.0304      0.014      2.238      0.026       0.004       0.057
basement            0.0216      0.011      1.946      0.052      -0.000       0.043
hotwaterheating     0.0849      0.022      3.941      0.000       0.043       0.127
airconditioning     0.0668      0.011      5.923      0.000       0.045       0.089
parking             0.0608      0.018      3.372      0.001       0.025       0.096
prefarea            0.0594      0.012      5.046      0.000       0.036       0.083
unfurnished        -0.0316      0.010     -3.096      0.002      -0.052      -0.012
==============================================================================
Omnibus:                       93.538   Durbin-Watson:                   2.092
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              303.844
Skew:                           1.090   Prob(JB):                     1.05e-66
Kurtosis:                       6.794   Cond. No.                         14.1
==============================================================================

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

In [53]:

```
# Calculate the VIFs again for the new model

vif = pd.DataFrame()
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
```

Out[53]:

	Features	VIF
1	bedrooms	6.59
4	mainroad	5.68
0	area	4.67
3	stories	2.69
9	parking	2.12
6	basement	2.01
8	airconditioning	1.77
2	bathrooms	1.67
10	prefarea	1.51
5	guestroom	1.47
11	unfurnished	1.40
7	hotwaterheating	1.14

In [54]:

```
# Dropping highly correlated variables and insignificant variables
X = X.drop('bedrooms', 1)
```

In [55]:

```
# Build a second fitted model
X_train_lm = sm.add_constant(X)

lr_3 = sm.OLS(y_train, X_train_lm).fit()
```

In [56]:

```
# Print the summary of the model

print(lr_3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.680
Model:                            OLS   Adj. R-squared:                  0.671
Method:                 Least Squares   F-statistic:                     71.31
Date:                Wed, 20 May 2020   Prob (F-statistic):           2.73e-84
Time:                        14:57:04   Log-Likelihood:                 380.96
No. Observations:                 381   AIC:                            -737.9
Df Residuals:                     369   BIC:                            -690.6
Df Model:                          11
Covariance Type:            nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
const               0.0357      0.015      2.421      0.016       0.007       0.065
area                0.2347      0.030      7.851      0.000       0.176       0.294
bathrooms           0.1965      0.022      9.132      0.000       0.154       0.239
stories             0.1178      0.018      6.654      0.000       0.083       0.153
mainroad            0.0488      0.014      3.423      0.001       0.021       0.077
guestroom           0.0301      0.014      2.211      0.028       0.003       0.057
basement            0.0239      0.011      2.183      0.030       0.002       0.045
hotwaterheating     0.0864      0.022      4.014      0.000       0.044       0.129
airconditioning     0.0665      0.011      5.895      0.000       0.044       0.089
parking             0.0629      0.018      3.501      0.001       0.028       0.098
prefarea            0.0596      0.012      5.061      0.000       0.036       0.083
unfurnished        -0.0323      0.010     -3.169      0.002      -0.052      -0.012
==============================================================================
Omnibus:                       97.661   Durbin-Watson:                   2.097
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              325.388
Skew:                           1.130   Prob(JB):                     2.20e-71
Kurtosis:                       6.923   Cond. No.                         10.6
==============================================================================

Warnings:
```

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

In [57]:

```python
# Calculate the VIFs again for the new model
vif = pd.DataFrame()
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
```

Out[57]:

	Features	VIF
3	mainroad	4.79
0	area	4.55
2	stories	2.23
8	parking	2.10
5	basement	1.87
7	airconditioning	1.76
1	bathrooms	1.61
9	prefarea	1.50
4	guestroom	1.46
10	unfurnished	1.33
6	hotwaterheating	1.12

In [58]:

```python
X = X.drop('basement', 1)
```

In [59]:

```python
# Build a fourth fitted model
X_train_lm = sm.add_constant(X)

lr_4 = sm.OLS(y_train, X_train_lm).fit()
```

In [60]:

```python
print(lr_4.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  price   R-squared:                       0.676
Model:                            OLS   Adj. R-squared:                  0.667
Method:                 Least Squares   F-statistic:                     77.18
Date:                Wed, 20 May 2020   Prob (F-statistic):           3.13e-84
Time:                        14:58:20   Log-Likelihood:                 378.51
No. Observations:                 381   AIC:                            -735.0
Df Residuals:                     370   BIC:                            -691.7
Df Model:                          10                                         
Covariance Type:            nonrobust                                         
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
const               0.0428      0.014      2.958      0.003       0.014       0.071
area                0.2335      0.030      7.772      0.000       0.174       0.293
bathrooms           0.2019      0.021      9.397      0.000       0.160       0.244
stories             0.1081      0.017      6.277      0.000       0.074       0.142
mainroad            0.0497      0.014      3.468      0.001       0.022       0.078
guestroom           0.0402      0.013      3.124      0.002       0.015       0.065
hotwaterheating     0.0876      0.022      4.051      0.000       0.045       0.130
airconditioning     0.0682      0.011      6.028      0.000       0.046       0.090
parking             0.0629      0.018      3.482      0.001       0.027       0.098
```

```
prefarea            0.0637      0.012      5.452      0.000       0.041       0.087
unfurnished        -0.0337      0.010     -3.295      0.001      -0.054      -0.014
==============================================================================
Omnibus:                       97.054   Durbin-Watson:                   2.099
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              322.034
Skew:                           1.124   Prob(JB):                     1.18e-70
Kurtosis:                       6.902   Cond. No.                         10.3
==============================================================================

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

In [61]:

```python
# Calculate the VIFs again for the new model
vif = pd.DataFrame()
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
```

Out[61]:

	Features	VIF
3	mainroad	4.55
0	area	4.54
2	stories	2.12
7	parking	2.10
6	airconditioning	1.75
1	bathrooms	1.58
8	prefarea	1.47
9	unfurnished	1.33
4	guestroom	1.30
5	hotwaterheating	1.12

In [62]:

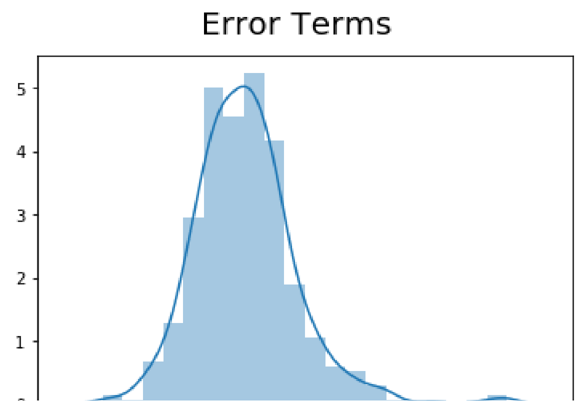
```python
y_train_price = lr_4.predict(X_train_lm)
```

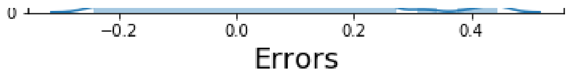
In [63]:

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

Out[63]:

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





predictions using final modal

In [64]:

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

df_test[num_vars] = scaler.transform(df_test[num_vars])
```

In [65]:

```
df_test.describe()
```

Out[65]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	aircondit
count	1.640000e+02	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	164.000000	164.(
mean	4.789686e+06	5228.695122	3.042683	1.317073	1.804878	0.865854	0.195122	0.347561	0.030488	0.:
std	1.987485e+06	2408.283816	0.737685	0.562162	0.828022	0.341853	0.397508	0.477654	0.172452	0.4
min	1.820000e+06	1650.000000	2.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.(
25%	3.395000e+06	3518.000000	3.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.(
50%	4.361000e+06	4787.500000	3.000000	1.000000	2.000000	1.000000	0.000000	0.000000	0.000000	0.(
75%	5.757500e+06	6352.500000	3.000000	2.000000	2.000000	1.000000	0.000000	1.000000	0.000000	1.(
max	1.225000e+07	16200.000000	5.000000	4.000000	4.000000	1.000000	1.000000	1.000000	1.000000	1.(

In [66]:

```
y_test = df_test.pop('price')
X_test = df_test
```

In [67]:

```
# Adding constant variable to test dataframe
X_test_m4 = sm.add_constant(X_test)
```

In [68]:

```
# Creating X_test_m4 dataframe by dropping variables from X_test_m4

X_test_m4 = X_test_m4.drop(["bedrooms", "semi-furnished", "basement"], axis = 1)
```

In [69]:

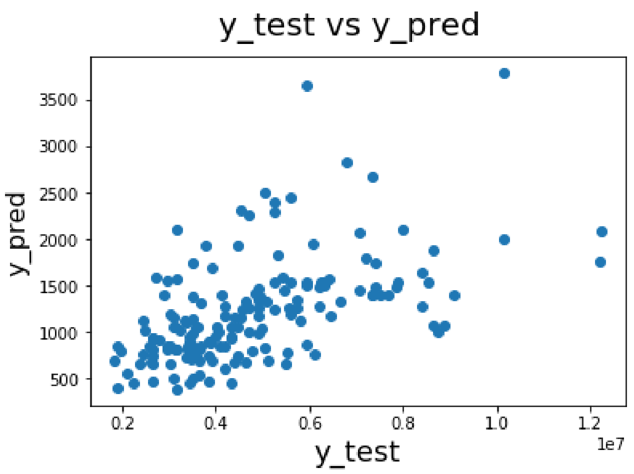
```
# Making predictions using the fourth model

y_pred_m4 = lr_4.predict(X_test_m4)
```

In [71]:

```
# Plotting y_test and y_pred to understand the spread

fig = plt.figure()
plt.scatter(y_test, y_pred_m4)
fig.suptitle('y_test vs y_pred', fontsize = 20)
plt.xlabel('y_test', fontsize = 18)
plt.ylabel('y_pred', fontsize = 16)
```

Out[71]:

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



overall we have a decent model but we also acknowledge that we could do better

In [ ]:

```

```