Housing Case Study

Problem Statement:

Consider a real estate company that has a dataset containing the prices of properties in the Delhi region. It wishes to use the data to optimise the sale prices of the properties based on important factors such as area, bedrooms, parking, etc.

Essentially, the company wants —

- To identify the variables affecting house prices, e.g. area, number of rooms, bathrooms, etc.
- To create a linear model that quantitatively relates house prices with variables such as number of rooms, area, number of bathrooms, etc.
- To know the accuracy of the model, i.e. how well these variables can predict house prices.

Importing and Understanding Data

In [54]:
```
import pandas as pd
import numpy as np
```

In [55]:
```
# Importing Housing.csv
housing = pd.read_csv('Housing.csv')
```

In [56]:
```
# Looking at the first five rows
housing.head()
```

Out[56]:

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

In [57]:
```
# What type of values are stored in the columns?
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
```

Data Preparation

- You can see that your dataset has many columns with values as 'Yes' or 'No'.
- We need to convert them to 1s and 0s, where 1 is a 'Yes' and 0 is a 'No'.

In [58]:
```
# Converting Yes to 1 and No to 0
housing['mainroad'] = housing['mainroad'].map({'yes': 1, 'no': 0})
housing['guestroom'] = housing['guestroom'].map({'yes': 1, 'no': 0})
housing['basement'] = housing['basement'].map({'yes': 1, 'no': 0})
housing['hotwaterheating'] = housing['hotwaterheating'].map({'yes': 1, 'no': 0
})
housing['airconditioning'] = housing['airconditioning'].map({'yes': 1, 'no': 0
})
housing['prefarea'] = housing['prefarea'].map({'yes': 1, 'no': 0})
```

In [59]:
```
# Now let's see the head
housing.head()
```

Out[59]:

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

The variable 'furnishingstatus' had three levels. We need to convert it to integer.

In [60]:
```
# Creating a dummy variable for 'furnishingstatus'
status = pd.get_dummies(housing['furnishingstatus'])
```

In [61]:
```
# The result has created three variables that are not needed.
status.head()
```

Out[61]:

	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 [62]:
```
# we don't need 3 columns.
# we can use drop_first = True to drop the first column from status df.
status = pd.get_dummies(housing['furnishingstatus'],drop_first=True)
```

In [63]:
```
#Adding the results to the master dataframe
housing = pd.concat([housing,status],axis=1)
```

In [64]:
```
# Now let's see the head of our dataframe.
housing.head()
```

Out[64]:

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

In [65]:
```
# Dropping furnishingstatus as we have created the dummies for it
housing.drop(['furnishingstatus'],axis=1,inplace=True)
```

In [66]:
```
# Now let's see the head of our dataframe.
housing.head()
```

Out[66]:

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

Creating a new variable

In [67]:
```
# Let us create the new metric and assign it to "areaperbedroom"
housing['areaperbedroom'] = housing['area']/housing['bedrooms']
```

In [68]:
```
# Metric:bathrooms per bedroom
housing['bbratio'] = housing['bathrooms']/housing['bedrooms']
```

```
In [69]: housing.head()
```

Out[69]:

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

Rescaling the Features

It is extremely important to rescale the variables so that they have a comparable scale. There are twocoon ways of rescaling

1. Normalisation (min-max scaling) and
2. standardisation (mean-o, sigma-1) Let's try normalisation

```
In [70]: #defining a normalisation function
         def normalize (x):
             return ( (x-np.mean(x))/ (max(x) - min(x)))


         # applying normalize ( ) to all columns
         housing = housing.apply(normalize)
```

Splitting Data into Training and Testing Sets

```
In [71]: housing.columns

Out[71]: Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
                'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
                'parking', 'prefarea', 'semi-furnished', 'unfurnished',
                'areaperbedroom', 'bbratio'],
               dtype='object')
```

In [72]:
```
# Putting feature variable to X
X = housing[['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'semi-furnished', 'unfurnished',
       'areaperbedroom', 'bbratio']]

# Putting response variable to y
y = housing['price']
```

In [73]:
```
#random_state is the seed used by the random number generator, it can be any i
nteger.
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7 ,test
_size = 0.3, random_state=100)
```

Building a linear model

In [74]:
```
import statsmodels.api as sm          # Importing statsmodels
X_train = sm.add_constant(X_train)    # Adding a constant column to our datafr
ame
# create a first fitted model
lm_1 = sm.OLS(y_train,X_train).fit()
```

In [75]:
```
#Let's see the summary of our first linear model
print(lm_1.summary())
```

```
                            OLS Regression Results
=============================================================================
=
Dep. Variable:                  price   R-squared:                       0.68
6
Model:                            OLS   Adj. R-squared:                  0.67
3
Method:                 Least Squares   F-statistic:                     53.1
2
Date:                Thu, 01 Mar 2018   Prob (F-statistic):           4.56e-8
2
Time:                        14:44:46   Log-Likelihood:                 384.4
0
No. Observations:                 381   AIC:                            -736.
8
Df Residuals:                     365   BIC:                            -673.
7
Df Model:                          15
Covariance Type:            nonrobust
=============================================================================
======
                      coef    std err          t      P>|t|      [0.025
0.975]
-----------------------------------------------------------------------------
------
const               0.0022      0.005      0.474      0.636      -0.007
0.011
area                0.5745      0.134      4.285      0.000       0.311
0.838
bedrooms           -0.0587      0.093     -0.632      0.528      -0.241
0.124
bathrooms           0.2336      0.126      1.849      0.065      -0.015
0.482
stories             0.1018      0.019      5.265      0.000       0.064
0.140
mainroad            0.0511      0.014      3.580      0.000       0.023
0.079
guestroom           0.0260      0.014      1.887      0.060      -0.001
0.053
basement            0.0208      0.011      1.877      0.061      -0.001
0.043
hotwaterheating     0.0875      0.022      4.048      0.000       0.045
0.130
airconditioning     0.0663      0.011      5.868      0.000       0.044
0.088
parking             0.0562      0.018      3.104      0.002       0.021
0.092
prefarea            0.0566      0.012      4.772      0.000       0.033
0.080
semi-furnished     -0.0008      0.012     -0.068      0.946      -0.024
0.022
unfurnished        -0.0323      0.013     -2.550      0.011      -0.057
-0.007
areaperbedroom     -0.3135      0.147     -2.139      0.033      -0.602
-0.025
bbratio             0.0439      0.104      0.421      0.674      -0.161
0.249
```

```
==============================================================================
=
Omnibus:                       87.283   Durbin-Watson:                   2.08
7
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              276.32
8
Skew:                           1.023   Prob(JB):                     9.91e-6
1
Kurtosis:                       6.636   Cond. No.                         47.
9
==============================================================================
=

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

Checking VIF

In [76]:
```
# UDF for calculating vif value
def vif_cal(input_data, dependent_col):
    vif_df = pd.DataFrame( columns = ['Var', 'Vif'])
    x_vars=input_data.drop([dependent_col], axis=1)
    xvar_names=x_vars.columns
    for i in range(0,xvar_names.shape[0]):
        y=x_vars[xvar_names[i]]
        x=x_vars[xvar_names.drop(xvar_names[i])]
        rsq=sm.OLS(y,x).fit().rsquared
        vif=round(1/(1-rsq),2)
        vif_df.loc[i] = [xvar_names[i], vif]
    return vif_df.sort_values(by = 'Vif', axis=0, ascending=False, inplace=Fal
se)
```

In [77]:
```
# Calculating Vif value
vif_cal(input_data=housing, dependent_col="price")
```

Out[77]:

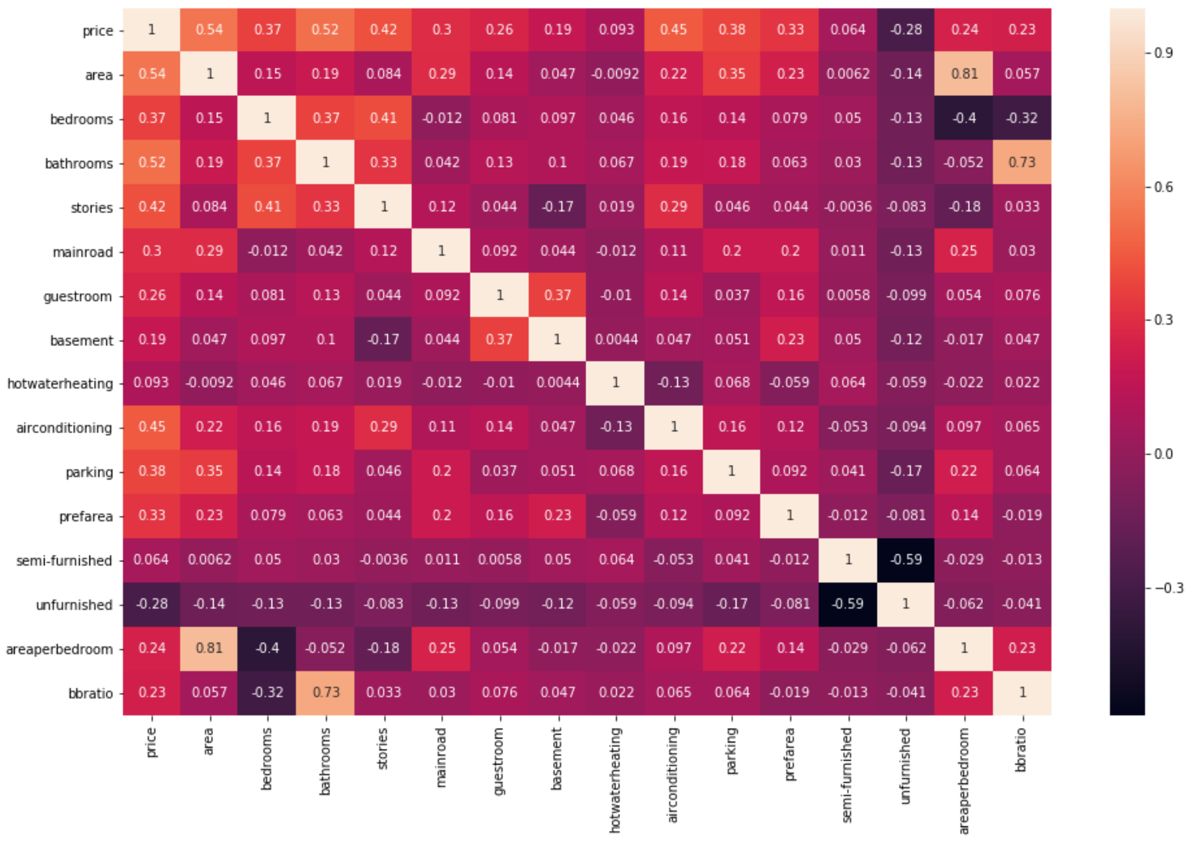
	Var	Vif
2	bathrooms	20.21
14	bbratio	19.04
13	areaperbedroom	17.59
0	area	16.00
1	bedrooms	9.11
12	unfurnished	1.68
11	semi-furnished	1.59
3	stories	1.51
6	basement	1.33
5	guestroom	1.23
9	parking	1.22
8	airconditioning	1.21
4	mainroad	1.17
10	prefarea	1.16
7	hotwaterheating	1.05

Correlation matrix

In [78]:
```
# Importing matplotlib and seaborn
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [79]:
```
# Let's see the correlation matrix
plt.figure(figsize = (16,10))     # Size of the figure
sns.heatmap(housing.corr(),annot = True)
```

Out[79]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x12903489be0>
```



Dropping the Variable and Updating the Model

In [80]:
```
# Dropping highly correlated variables and insignificant variables
X_train = X_train.drop('bbratio', 1)
```

In [81]:
```
# Create a second fitted model
lm_2 = sm.OLS(y_train,X_train).fit()
```

In [82]:
```
#Let's see the summary of our second linear model
print(lm_2.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                  price   R-squared:                       0.68
6
Model:                            OLS   Adj. R-squared:                  0.67
4
Method:                 Least Squares   F-statistic:                     57.0
3
Date:                Thu, 01 Mar 2018   Prob (F-statistic):           6.46e-8
3
Time:                        14:44:48   Log-Likelihood:                 384.3
1
No. Observations:                 381   AIC:                            -738.
6
Df Residuals:                     366   BIC:                            -679.
5
Df Model:                          14
Covariance Type:            nonrobust
==============================================================================
======
                      coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
------
const               0.0022      0.005      0.482      0.630      -0.007
0.011
area                0.5483      0.119      4.622      0.000       0.315
0.782
bedrooms           -0.0845      0.070     -1.209      0.227      -0.222
0.053
bathrooms           0.2850      0.033      8.686      0.000       0.220
0.350
stories             0.1022      0.019      5.301      0.000       0.064
0.140
mainroad            0.0509      0.014      3.568      0.000       0.023
0.079
guestroom           0.0265      0.014      1.941      0.053      -0.000
0.053
basement            0.0210      0.011      1.898      0.058      -0.001
0.043
hotwaterheating     0.0866      0.021      4.031      0.000       0.044
0.129
airconditioning     0.0662      0.011      5.871      0.000       0.044
0.088
parking             0.0563      0.018      3.119      0.002       0.021
0.092
prefarea            0.0563      0.012      4.760      0.000       0.033
0.079
semi-furnished     -0.0009      0.012     -0.077      0.939      -0.024
0.022
unfurnished        -0.0323      0.013     -2.554      0.011      -0.057
-0.007
areaperbedroom     -0.2840      0.129     -2.208      0.028      -0.537
-0.031
==============================================================================
=
Omnibus:                       88.466   Durbin-Watson:                   2.08
5
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              282.79
4
Skew:                           1.034   Prob(JB):                      3.91e-6
2
Kurtosis:                       6.679   Cond. No.                         39.
7
==============================================================================
=

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

Dropping the Variable and Updating the Model

In [83]:
```python
# Calculating Vif value
vif_cal(input_data=housing.drop(["bbratio"], axis=1), dependent_col="price")
```

Out[83]:

	Var	Vif
13	areaperbedroom	14.14
0	area	12.84
1	bedrooms	4.99
12	unfurnished	1.68
11	semi-furnished	1.59
3	stories	1.50
6	basement	1.32
2	bathrooms	1.29
5	guestroom	1.22
9	parking	1.22
8	airconditioning	1.21
4	mainroad	1.17
10	prefarea	1.16
7	hotwaterheating	1.04

In [84]:
```python
# Dropping highly correlated variables and insignificant variables
X_train = X_train.drop('bedrooms', 1)
```

In [85]:
```
# Create a third fitted model
lm_3 = sm.OLS(y_train,X_train).fit()
```

In [86]:
```
#Let's see the summary of our third linear model
print(lm_3.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                  price   R-squared:                       0.68
4
Model:                            OLS   Adj. R-squared:                  0.67
3
Method:                 Least Squares   F-statistic:                     61.2
3
Date:                Thu, 01 Mar 2018   Prob (F-statistic):           1.66e-8
3
Time:                        14:44:49   Log-Likelihood:                 383.5
5
No. Observations:                 381   AIC:                            -739.
1
Df Residuals:                     367   BIC:                            -683.
9
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
======
                      coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
------
const               0.0021      0.005      0.459      0.647      -0.007
0.011
area                0.4326      0.070      6.164      0.000       0.295
0.571
bathrooms           0.2814      0.033      8.606      0.000       0.217
0.346
stories             0.1005      0.019      5.224      0.000       0.063
0.138
mainroad            0.0515      0.014      3.611      0.000       0.023
0.080
guestroom           0.0285      0.014      2.101      0.036       0.002
0.055
basement            0.0201      0.011      1.822      0.069      -0.002
0.042
hotwaterheating     0.0850      0.021      3.963      0.000       0.043
0.127
airconditioning     0.0667      0.011      5.909      0.000       0.044
0.089
parking             0.0573      0.018      3.175      0.002       0.022
0.093
prefarea            0.0576      0.012      4.895      0.000       0.034
0.081
semi-furnished   9.202e-06      0.012      0.001      0.999      -0.023
0.023
unfurnished        -0.0313      0.013     -2.478      0.014      -0.056
-0.006
areaperbedroom     -0.1516      0.068     -2.242      0.026      -0.285
-0.019
==============================================================================
=
Omnibus:                       88.924   Durbin-Watson:                   2.08
5
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              283.14
0
Skew:                           1.041   Prob(JB):                     3.29e-6
2
Kurtosis:                       6.674   Cond. No.                         20.
4
==============================================================================
=

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

In [87]:
```
# Calculating Vif value
vif_cal(input_data=housing.drop(["bedrooms","bbratio"], axis=1), dependent_col
="price")
```

Out[87]:

	Var	Vif
0	area	4.21
12	areaperbedroom	3.88
11	unfurnished	1.67
10	semi-furnished	1.58
2	stories	1.49
5	basement	1.32
1	bathrooms	1.29
8	parking	1.22
4	guestroom	1.21
7	airconditioning	1.21
3	mainroad	1.17
9	prefarea	1.15
6	hotwaterheating	1.04

Dropping the Variable and Updating the Model

In [88]:
```
# # Dropping highly correlated variables and insignificant variables
X_train = X_train.drop('areaperbedroom', 1)
```

In [89]:
```
# Create a fourth fitted model
lm_4 = sm.OLS(y_train,X_train).fit()
```

In [90]:
```
#Let's see the summary of our fourth linear model
print(lm_4.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                  price   R-squared:                       0.68
0
Model:                            OLS   Adj. R-squared:                  0.67
0
Method:                 Least Squares   F-statistic:                     65.2
0
Date:                Thu, 01 Mar 2018   Prob (F-statistic):          2.35e-8
3
Time:                        14:44:49   Log-Likelihood:                 380.9
6
No. Observations:                 381   AIC:                            -735.
9
Df Residuals:                     368   BIC:                            -684.
7
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
======
                      coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
------
const               0.0013      0.005      0.287      0.775      -0.008
0.010
area                0.3008      0.039      7.799      0.000       0.225
0.377
bathrooms           0.2947      0.032      9.114      0.000       0.231
0.358
stories             0.1178      0.018      6.643      0.000       0.083
0.153
mainroad            0.0488      0.014      3.419      0.001       0.021
0.077
guestroom           0.0301      0.014      2.207      0.028       0.003
0.057
basement            0.0239      0.011      2.179      0.030       0.002
0.045
hotwaterheating     0.0864      0.022      4.007      0.000       0.044
0.129
airconditioning     0.0666      0.011      5.870      0.000       0.044
0.089
parking             0.0629      0.018      3.495      0.001       0.027
0.098
prefarea            0.0597      0.012      5.055      0.000       0.036
0.083
semi-furnished      0.0008      0.012      0.067      0.947      -0.022
0.024
unfurnished        -0.0318      0.013     -2.504      0.013      -0.057
-0.007
==============================================================================
=
Omnibus:                       97.809   Durbin-Watson:                   2.09
7
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              326.48
5
```

```
Skew:                           1.131   Prob(JB):                    1.27e-7
1
Kurtosis:                       6.930   Cond. No.                        8.5
2
=============================================================================
=

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

In [91]:
```
# Calculating Vif value
vif_cal(input_data=housing.drop(["bedrooms","bbratio","areaperbedroom"], axis=
1), dependent_col="price")
```

Out[91]:

	Var	Vif
11	unfurnished	1.67
10	semi-furnished	1.58
0	area	1.32
2	stories	1.30
5	basement	1.30
1	bathrooms	1.22
4	guestroom	1.21
7	airconditioning	1.21
8	parking	1.21
3	mainroad	1.16
9	prefarea	1.15
6	hotwaterheating	1.04

Dropping the Variable and Updating the Model

In [92]:
```
# # Dropping highly correlated variables and insignificant variables
X_train = X_train.drop('semi-furnished', 1)
```

In [93]:
```
# Create a fifth fitted model
lm_5 = sm.OLS(y_train,X_train).fit()
```

In [94]:
```
#Let's see the summary of our fifth linear model
print(lm_5.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                  price   R-squared:                       0.68
0
Model:                            OLS   Adj. R-squared:                  0.67
1
Method:                 Least Squares   F-statistic:                     71.3
1
Date:                Thu, 01 Mar 2018   Prob (F-statistic):          2.73e-8
4
Time:                        14:44:50   Log-Likelihood:                 380.9
6
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               0.0013      0.005      0.286      0.775      -0.008
0.010
area                0.3006      0.038      7.851      0.000       0.225
0.376
bathrooms           0.2947      0.032      9.132      0.000       0.231
0.358
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:                   2.09
7
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              325.38
8
Skew:                           1.130   Prob(JB):                     2.20e-7
1
```

```
Kurtosis:                       6.923   Cond. No.                       8.4
6
==============================================================================
=

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

In [95]:
```
# Calculating Vif value
vif_cal(input_data=housing.drop(["bedrooms","bbratio","areaperbedroom","semi-f
urnished"], axis=1), dependent_col="price")
```

Out[95]:

	Var	Vif
0	area	1.32
2	stories	1.30
5	basement	1.30
1	bathrooms	1.22
4	guestroom	1.21
8	parking	1.21
7	airconditioning	1.20
3	mainroad	1.15
9	prefarea	1.15
10	unfurnished	1.07
6	hotwaterheating	1.04

Dropping the Variable and Updating the Model

In [96]:
```
# # Dropping highly correlated variables and insignificant variables
X_train = X_train.drop('basement', 1)
```

In [97]:
```
# Create a sixth fitted model
lm_6 = sm.OLS(y_train,X_train).fit()
```

In [98]:
```
#Let's see the summary of our sixth linear model
print(lm_6.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                  price   R-squared:                       0.67
6
Model:                            OLS   Adj. R-squared:                  0.66
7
Method:                 Least Squares   F-statistic:                     77.1
8
Date:                Thu, 01 Mar 2018   Prob (F-statistic):           3.13e-8
4
Time:                        14:44:50   Log-Likelihood:                 378.5
1
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               0.0015      0.005      0.320      0.749      -0.008
0.011
area                0.2990      0.038      7.772      0.000       0.223
0.375
bathrooms           0.3028      0.032      9.397      0.000       0.239
0.366
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:                   2.09
9
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              322.03
4
Skew:                           1.124   Prob(JB):                     1.18e-7
0
Kurtosis:                       6.902   Cond. No.                         8.4
5
```

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

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

In [99]:
```
# Calculating Vif value
vif_cal(input_data=housing.drop(["bedrooms","bbratio","areaperbedroom","semi-f
urnished","basement"], axis=1), dependent_col="price")
```

Out[99]:

	Var	Vif
0	area	1.31
2	stories	1.22
7	parking	1.21
1	bathrooms	1.20
6	airconditioning	1.20
3	mainroad	1.15
8	prefarea	1.10
4	guestroom	1.07
9	unfurnished	1.06
5	hotwaterheating	1.04

Assessment question

Design four models by dropping all the variables one by one with high vif (>5). Then, compare the results.

Making Predictions Using the Final Model

Prediction with Model 6

In [100]:
```
# Adding  constant variable to test dataframe
X_test_m6 = sm.add_constant(X_test)
```

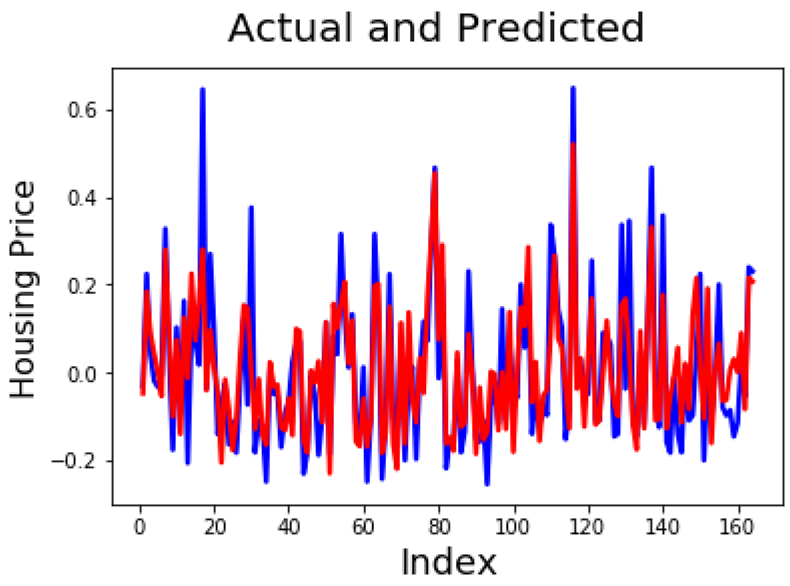
In [101]:
```
# Creating X_test_m6 dataframe by dropping variables from X_test_m6
X_test_m6 = X_test_m6.drop(["bedrooms","bbratio","areaperbedroom","semi-furnis
hed","basement"], axis=1)
```

```
In [102]:  # Making predictions
           y_pred_m6 = lm_6.predict(X_test_m6)
```

Model Evaluation

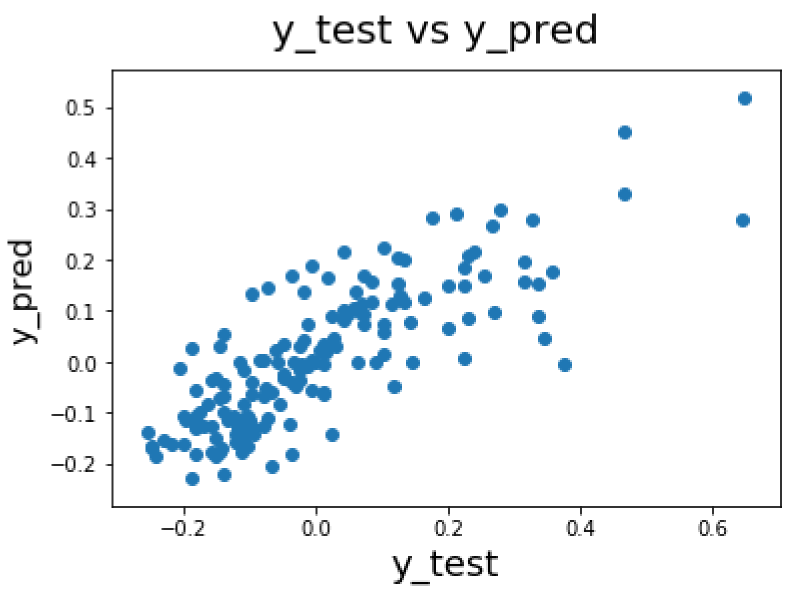
```
In [103]:  # Actual vs Predicted
           c = [i for i in range(1,165,1)]
           fig = plt.figure()
           plt.plot(c,y_test, color="blue", linewidth=2.5, linestyle="-")     #Plotting A
           ctual
           plt.plot(c,y_pred_m6, color="red",  linewidth=2.5, linestyle="-")  #Plotting p
           redicted
           fig.suptitle('Actual and Predicted', fontsize=20)              # Plot heading
           plt.xlabel('Index', fontsize=18)                               # X-label
           plt.ylabel('Housing Price', fontsize=16)                       # Y-label

Out[103]:  Text(0,0.5,'Housing Price')
```



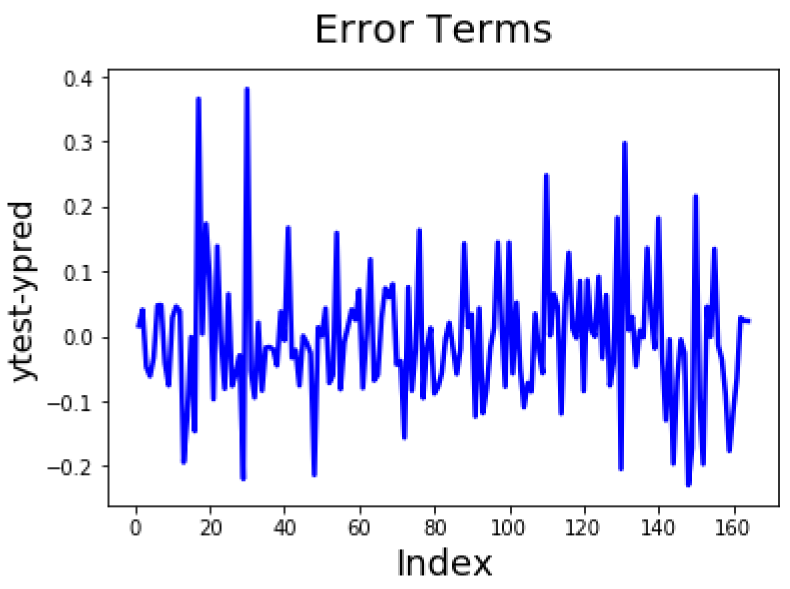
In [104]:
```
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test,y_pred_m6)
fig.suptitle('y_test vs y_pred', fontsize=20)              # Plot heading
plt.xlabel('y_test', fontsize=18)                          # X-label
plt.ylabel('y_pred', fontsize=16)                          # Y-label
```

Out[104]:
```
Text(0,0.5,'y_pred')
```



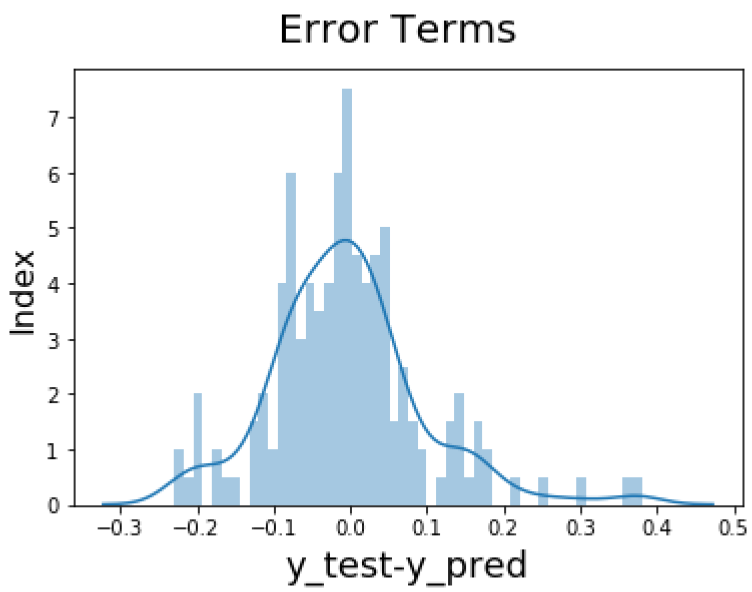
In [105]:
```
# Error terms
fig = plt.figure()
c = [i for i in range(1,165,1)]
plt.plot(c,y_test-y_pred_m6, color="blue", linewidth=2.5, linestyle="-")
fig.suptitle('Error Terms', fontsize=20)              # Plot heading
plt.xlabel('Index', fontsize=18)                      # X-label
plt.ylabel('ytest-ypred', fontsize=16)                # Y-label
```

Out[105]:
```
Text(0,0.5,'ytest-ypred')
```



```
In [106]: # Plotting the error terms to understand the distribution.
          fig = plt.figure()
          sns.distplot((y_test-y_pred_m6),bins=50)
          fig.suptitle('Error Terms', fontsize=20)                  # Plot heading
          plt.xlabel('y_test-y_pred', fontsize=18)                  # X-label
          plt.ylabel('Index', fontsize=16)                          # Y-label

Out[106]: Text(0,0.5,'Index')
```



```
In [107]: import numpy as np
          from sklearn import metrics
          print('RMSE :', np.sqrt(metrics.mean_squared_error(y_test, y_pred_m6)))

          RMSE : 0.100010923368
```