house-price-prediction

October 3, 2024

1 Import Libraries

[2]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
# visualization
import matplotlib.pyplot as plt
import seaborn as sns
# model building
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

2 Import Dataset

[4]:
```
#data = pd.read_csv("/kaggle/input/housing-dataset.csv")
housing = pd.DataFrame(pd.read_csv("Housing.csv"))
housing.head()
```

[4]:

	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

3 Preparing the Data

[6]:
```
# Getting the dimensions of the data
print(f"Rows and Columns of data is :{housing.shape}")
```

```
Rows and Columns of data is :(545, 13)
```

[7]:
```
# Getting the null values of the data
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

[8]:
```
housing.describe(include = 'all')
```

[8]:

	price	area	bedrooms	bathrooms	stories	\
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	
unique	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	
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	\

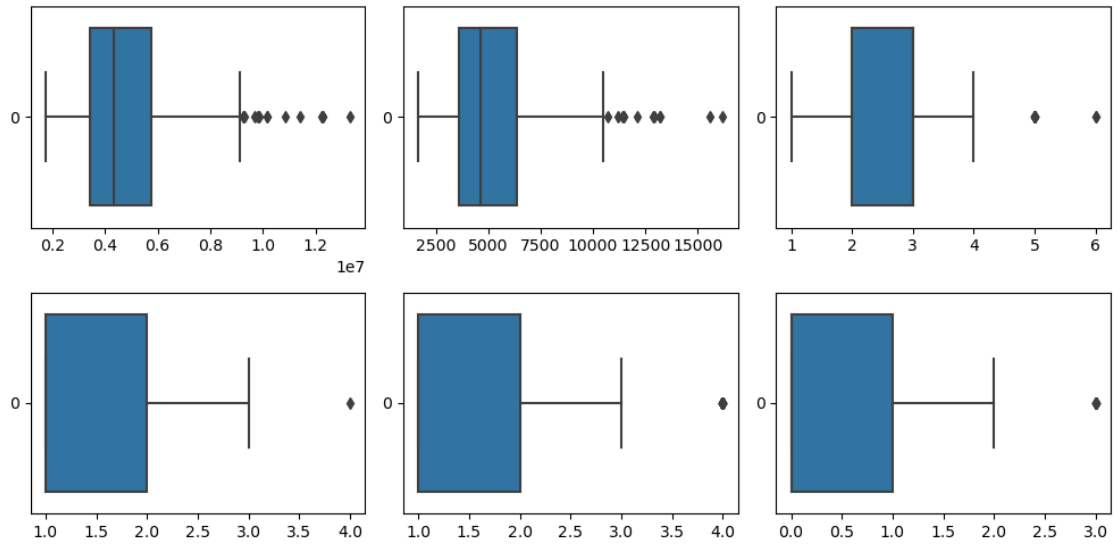
```
count       545       545      545             545             545
unique        2         2        2               2               2
top         yes        no       no              no              no
freq        468       448      354             520             373
mean        NaN       NaN      NaN             NaN             NaN
std         NaN       NaN      NaN             NaN             NaN
min         NaN       NaN      NaN             NaN             NaN
25%         NaN       NaN      NaN             NaN             NaN
50%         NaN       NaN      NaN             NaN             NaN
75%         NaN       NaN      NaN             NaN             NaN
max         NaN       NaN      NaN             NaN             NaN

            parking prefarea furnishingstatus
count   545.000000      545              545
unique         NaN        2                3
top            NaN       no   semi-furnished
freq           NaN      417              227
mean      0.693578      NaN              NaN
std       0.861586      NaN              NaN
min       0.000000      NaN              NaN
25%       0.000000      NaN              NaN
50%       0.000000      NaN              NaN
75%       1.000000      NaN              NaN
max       3.000000      NaN              NaN
```

[9]:
```
# Checking the null values
housing.isnull().sum()
```

[9]:
```
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
```

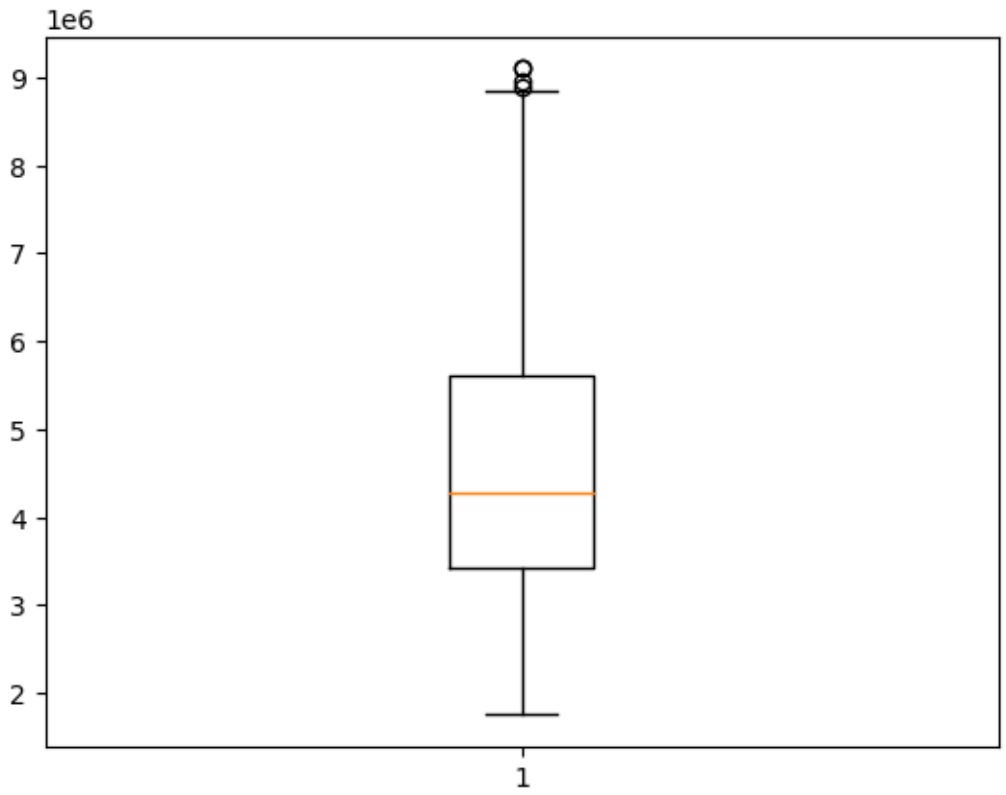
[10]:
```
#Outlier Analysis
fig,axs = plt.subplots(2,3, figsize = (10,5))
plt1 = sns.boxplot(housing['price'], ax = axs[0,0], orient="h")
plt2 = sns.boxplot(housing['area'], ax = axs[0,1], orient="h")
```

```
plt3 = sns.boxplot(housing['bedrooms'], ax = axs[0,2], orient="h")
plt4 = sns.boxplot(housing['bathrooms'], ax = axs[1,0], orient="h")
plt5 = sns.boxplot(housing['stories'], ax = axs[1,1], orient="h")
plt6 = sns.boxplot(housing['parking'], ax = axs[1,2], orient="h")
plt.tight_layout()
```

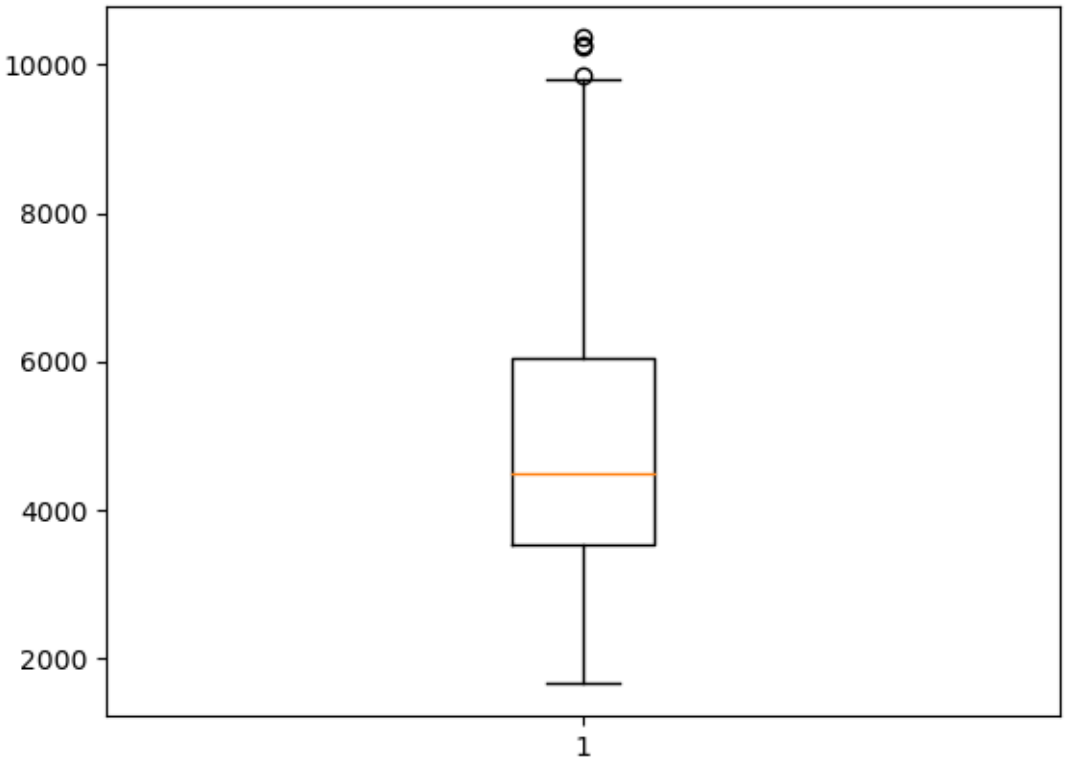


4 Cleaning the Data

[12]:
```
# outlier treatment for price
Q1 = housing.price.quantile(0.25)
Q3 = housing.price.quantile(0.75)
IQR = Q3 - Q1
housing = housing[(housing.price >= Q1 - 1.5*IQR) & (housing.price <= Q3 + 1.
↪5*IQR)]
plt.boxplot(housing.price)
plt.show()
```



[13]:
```
#Outlier Treatment or area
Q1 = housing.area.quantile(0.25)
Q3 = housing.area.quantile(0.75)
IQR = Q3-Q1
housing = housing[(housing.area >= Q1 - 1.5*IQR) & (housing.area <= Q3 + 1.
 ↪5*IQR)]
plt.boxplot(housing.area)
plt.show()
```



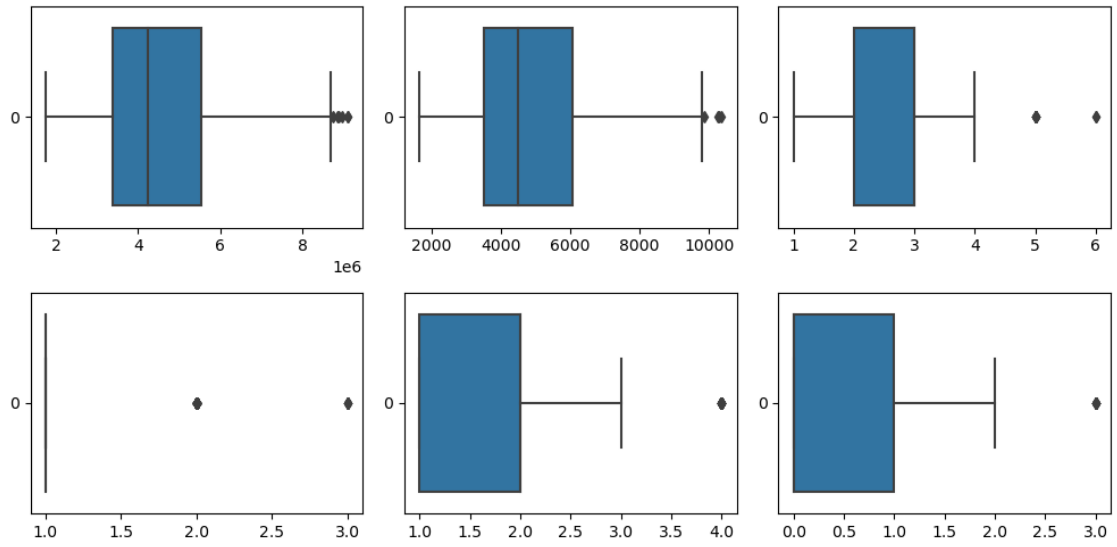
[14]:
```
housing = housing.reset_index()
housing.head()
```

[14]:

	index	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	15	9100000	6000	4	1	2	yes	no	yes	no	no	2	no	semi-furnished
1	16	9100000	6600	4	2	2	yes	yes	yes	no	yes	1	yes	unfurnished
2	17	8960000	8500	3	2	4	yes	no	no	no	yes	2	no	furnished
3	18	8890000	4600	3	2	2	yes	yes	no	no	yes	2	no	furnished
4	19	8855000	6420	3	2	2	yes	no	no	no	yes	1	yes	semi-furnished

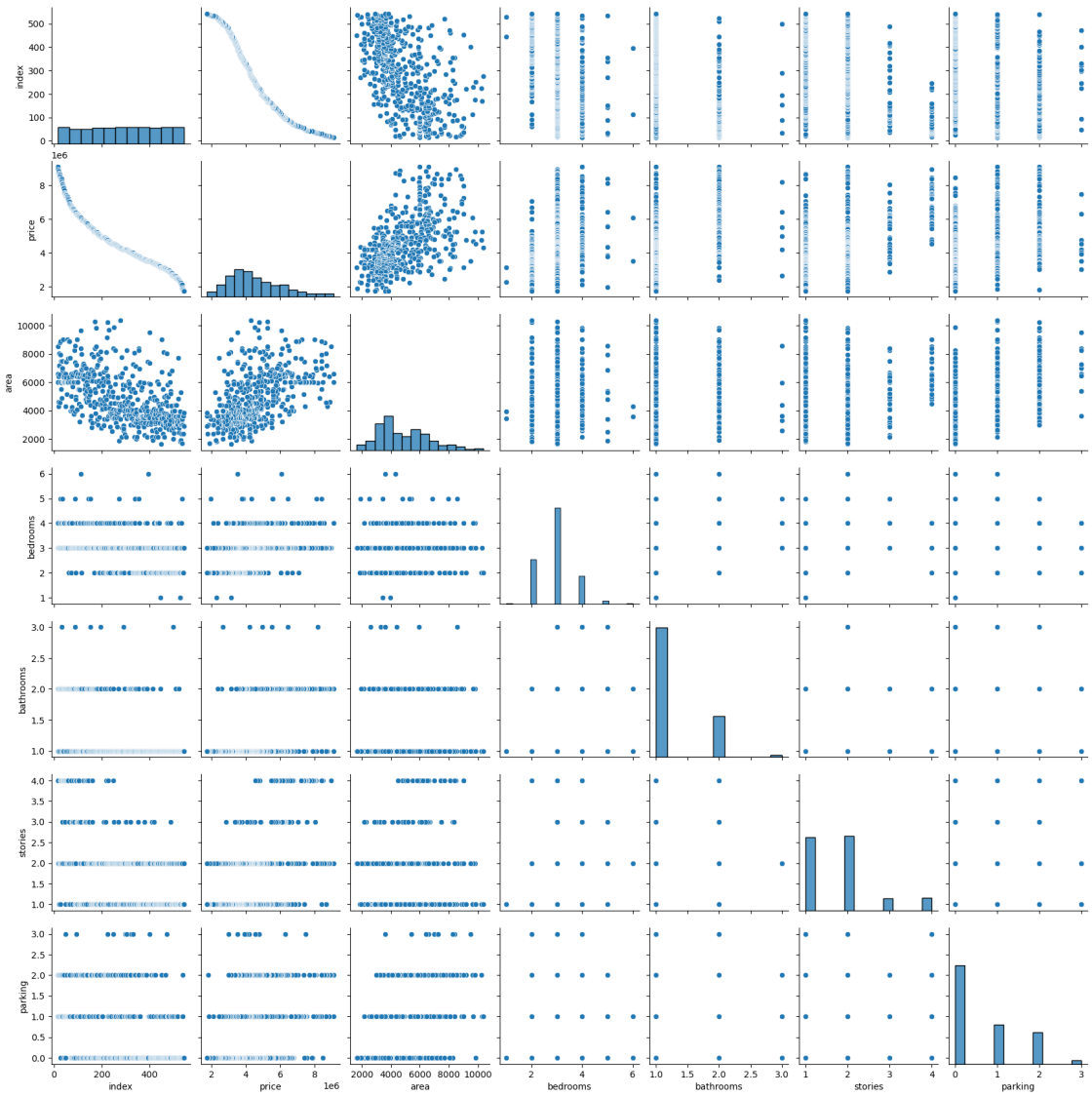
[15]:
```
# Outlier Analysis
#Outlier Analysis
fig,axs = plt.subplots(2,3, figsize = (10,5))
plt1 = sns.boxplot(housing['price'], ax = axs[0,0], orient="h")
plt2 = sns.boxplot(housing['area'], ax = axs[0,1], orient="h")
```

```
plt3 = sns.boxplot(housing['bedrooms'], ax = axs[0,2], orient="h")
plt4 = sns.boxplot(housing['bathrooms'], ax = axs[1,0], orient="h")
plt5 = sns.boxplot(housing['stories'], ax = axs[1,1], orient="h")
plt6 = sns.boxplot(housing['parking'], ax = axs[1,2], orient="h")
plt.tight_layout()
```



[16]:
```
#Pairplot of numeric varialbles
sns.pairplot(housing)
plt.show()
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```



[17]:
```
#List of binary variables

binlist = ['mainroad', 'guestroom', 'basement', 'hotwaterheating',␣
 ↪'airconditioning', 'prefarea']

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

binlist = housing[binlist].apply(binary_map)
binlist.head()
```

[17]:

	mainroad	guestroom	basement	hotwaterheating	airconditioning	prefarea
0	1	0	1	0	0	0

```
1          1          1          1                  0                 1          1
2          1          0          0                  0                 1          0
3          1          1          0                  0                 1          0
4          1          0          0                  0                 1          1
```

[18]:
```
# Removing the categorical variables from the original housing dataframe
housing.drop(['mainroad', 'guestroom', 'basement', 'hotwaterheating',␣
 ↪'airconditioning', 'prefarea'], axis = 1, inplace = True)
housing.head()
```

[18]:
```
   index     price  area  bedrooms  bathrooms  stories  parking  \
0     15  9100000  6000         4          1        2        2
1     16  9100000  6600         4          2        2        1
2     17  8960000  8500         3          2        4        2
3     18  8890000  4600         3          2        2        2
4     19  8855000  6420         3          2        2        1

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

[19]:
```
# Adding the results to the original housing dataframe
housing = pd.concat([housing, binlist], axis = 1)
housing.head()
```

[19]:
```
   index     price  area  bedrooms  bathrooms  stories  parking  \
0     15  9100000  6000         4          1        2        2
1     16  9100000  6600         4          2        2        1
2     17  8960000  8500         3          2        4        2
3     18  8890000  4600         3          2        2        2
4     19  8855000  6420         3          2        2        1

  furnishingstatus  mainroad  guestroom  basement  hotwaterheating  \
0   semi-furnished         1          0         1                0
1      unfurnished         1          1         1                0
2        furnished         1          0         0                0
3        furnished         1          1         0                0
4   semi-furnished         1          0         0                0

   airconditioning  prefarea
0                0         0
1                1         1
2                1         0
3                1         0
```

```
4                1         1
```

[20]:
```
# Using pd.get_dummies() to one-hot encode the 'furnishingstatus' column

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

[20]:
```
   furnished  semi-furnished  unfurnished
0          0               1            0
1          0               0            1
2          1               0            0
3          1               0            0
4          0               1            0
```

[21]:
```
# Adding the results to the original housing dataframe
housing = pd.concat([housing, status], axis = 1)
housing.head()
```

[21]:
```
   index     price  area  bedrooms  bathrooms  stories  parking  \
0     15  9100000  6000         4          1        2        2
1     16  9100000  6600         4          2        2        1
2     17  8960000  8500         3          2        4        2
3     18  8890000  4600         3          2        2        2
4     19  8855000  6420         3          2        2        1

  furnishingstatus  mainroad  guestroom  basement  hotwaterheating  \
0   semi-furnished         1          0         1                0
1      unfurnished         1          1         1                0
2        furnished         1          0         0                0
3        furnished         1          1         0                0
4   semi-furnished         1          0         0                0

   airconditioning  prefarea  furnished  semi-furnished  unfurnished
0                0         0          0               1            0
1                1         1          0               0            1
2                1         0          1               0            0
3                1         0          1               0            0
4                1         1          0               1            0
```

[22]:
```
# Removing the 'furnishingstatus' column in dataframe

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

[22]:
```
   index     price  area  bedrooms  bathrooms  stories  parking  mainroad  \
0     15  9100000  6000         4          1        2        2         1
1     16  9100000  6600         4          2        2        1         1
```

```
2     17  8960000  8500         3          2        4        2         1
3     18  8890000  4600         3          2        2        2         1
4     19  8855000  6420         3          2        2        1         1

   guestroom  basement  hotwaterheating  airconditioning  prefarea  furnished  \
0          0         1                0                0         0          0
1          1         1                0                1         1          0
2          0         0                0                1         0          1
3          1         0                0                1         0          1
4          0         0                0                1         1          0

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

[23]:
```
# Identify the bool columns
bool_columns = housing.select_dtypes(include='bool').columns

# Convert the bool columns to int type
housing[bool_columns] = housing[bool_columns].astype(int)
housing.head()
```

[23]:
```
   index    price  area  bedrooms  bathrooms  stories  parking  mainroad  \
0     15  9100000  6000         4          1        2        2         1
1     16  9100000  6600         4          2        2        1         1
2     17  8960000  8500         3          2        4        2         1
3     18  8890000  4600         3          2        2        2         1
4     19  8855000  6420         3          2        2        1         1

   guestroom  basement  hotwaterheating  airconditioning  prefarea  furnished  \
0          0         1                0                0         0          0
1          1         1                0                1         1          0
2          0         0                0                1         0          1
3          1         0                0                1         0          1
4          0         0                0                1         1          0

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

5 Splitting of Dataset into train and test

[25]:
```
from sklearn.model_selection import train_test_split

np.random.seed(0)
x_train,x_test = train_test_split(housing,test_size = 0.2,random_state = 100)
```

6 Model Building

[27]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Apply Normalization for higher values and except yes or no and dummy variables

norm_vars = ['price', 'area' ,'bedrooms','bathrooms','stories','parking']
x_train[norm_vars] = scaler.fit_transform(x_train[norm_vars])
x_test[norm_vars] = scaler.transform(x_test[norm_vars])
```

[28]:
```
x_train.head()
```

[28]:

	index	price	area	bedrooms	bathrooms	stories	parking	\
412	440	0.201905	0.224018	0.6	0.0	0.333333	0.000000	
332	359	0.266667	0.219400	0.4	0.0	0.000000	0.333333	
423	451	0.190476	0.583141	0.2	0.0	0.000000	0.000000	
387	415	0.223810	0.356236	0.4	0.0	0.333333	0.333333	
153	177	0.475238	0.502309	0.4	0.0	0.000000	0.000000	

	mainroad	guestroom	basement	hotwaterheating	airconditioning	\
412	1	0	1	0	0	
332	1	0	0	0	0	
423	1	0	0	0	0	
387	1	1	1	0	1	
153	1	0	1	0	0	

	prefarea	furnished	semi-furnished	unfurnished
412	0	0	0	1
332	0	0	0	1
423	0	0	1	0
387	0	1	0	0
153	1	0	1	0

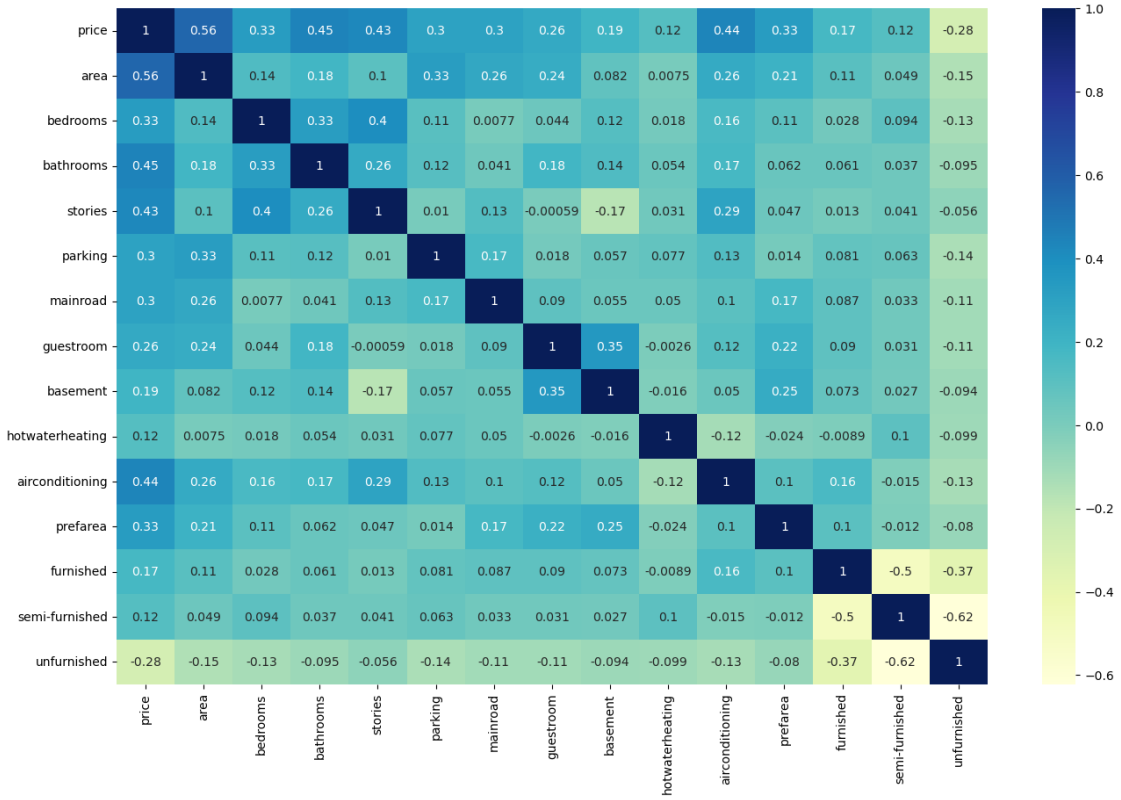
[29]:
```
x_train.describe()
```

[29]:

	index	price	area	bedrooms	bathrooms	stories	\
count	413.000000	413.000000	413.000000	413.000000	413.000000	413.000000	

```
mean   284.440678    0.380071    0.360160    0.387893    0.128329    0.259080
std    151.272690    0.213500    0.200796    0.149915    0.229492    0.280384
min     15.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%    154.000000    0.227619    0.207852    0.200000    0.000000    0.000000
50%    294.000000    0.333333    0.306236    0.400000    0.000000    0.333333
75%    414.000000    0.514286    0.496536    0.400000    0.000000    0.333333
max    544.000000    1.000000    1.000000    1.000000    1.000000    1.000000

          parking    mainroad   guestroom   basement  hotwaterheating  \
count  413.000000  413.000000  413.000000  413.00000       413.000000
mean     0.217111    0.871671    0.162228    0.35109         0.046005
std      0.281482    0.334862    0.369107    0.47789         0.209750
min      0.000000    0.000000    0.000000    0.00000         0.000000
25%      0.000000    1.000000    0.000000    0.00000         0.000000
50%      0.000000    1.000000    0.000000    0.00000         0.000000
75%      0.333333    1.000000    0.000000    1.00000         0.000000
max      1.000000    1.000000    1.000000    1.00000         1.000000

       airconditioning    prefarea   furnished  semi-furnished  unfurnished
count       413.000000  413.000000  413.000000      413.000000   413.000000
mean          0.292978    0.200969    0.227603        0.457627     0.314770
std           0.455681    0.401211    0.419793        0.498806     0.464987
min           0.000000    0.000000    0.000000        0.000000     0.000000
25%           0.000000    0.000000    0.000000        0.000000     0.000000
50%           0.000000    0.000000    0.000000        0.000000     0.000000
75%           1.000000    0.000000    0.000000        1.000000     1.000000
max           1.000000    1.000000    1.000000        1.000000     1.000000
```

[30]:
```python
# Checking the correlation between the variables

x_train.drop('index', axis =1, inplace = True)
plt.figure(figsize = (16, 10))
sns.heatmap(x_train.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



[31]:
```
x_train.head()
```

[31]:

	price	area	bedrooms	bathrooms	stories	parking	mainroad	\
412	0.201905	0.224018	0.6	0.0	0.333333	0.000000	1	
332	0.266667	0.219400	0.4	0.0	0.000000	0.333333	1	
423	0.190476	0.583141	0.2	0.0	0.000000	0.000000	1	
387	0.223810	0.356236	0.4	0.0	0.333333	0.333333	1	
153	0.475238	0.502309	0.4	0.0	0.000000	0.000000	1	

	guestroom	basement	hotwaterheating	airconditioning	prefarea	\
412	0	1	0	0	0	
332	0	0	0	0	0	
423	0	0	0	0	0	
387	1	1	0	1	0	
153	0	1	0	0	1	

	furnished	semi-furnished	unfurnished
412	0	0	1
332	0	0	1
423	0	1	0
387	1	0	0
153	0	1	0

[32]:
```
# Assigning the input and output variables

y_train = x_train.pop('price')
x_train.head()
```

[32]:

	area	bedrooms	bathrooms	stories	parking	mainroad	guestroom	basement	hotwaterheating	airconditioning	prefarea	furnished	semi-furnished	unfurnished
412	0.224018	0.6	0.0	0.333333	0.000000	1	0	1	0	0	0	0	0	1
332	0.219400	0.4	0.0	0.000000	0.333333	1	0	0	0	0	0	0	0	1
423	0.583141	0.2	0.0	0.000000	0.000000	1	0	0	0	0	0	0	1	0
387	0.356236	0.4	0.0	0.333333	0.333333	1	1	1	0	1	0	1	0	0
153	0.502309	0.4	0.0	0.000000	0.000000	1	0	1	0	0	1	0	1	0

[33]:
```
y_train.head()
```

[33]:
```
412    0.201905
332    0.266667
423    0.190476
387    0.223810
153    0.475238
Name: price, dtype: float64
```

[34]:
```
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE #RFE stands for Recursive Feature␣
 ↪Elimination

model = LinearRegression()
model.fit(x_train,y_train)
```

[34]:
```
LinearRegression()
```

[35]:
```
model.coef_
```

[35]:
```
array([3.44304446e-01, 2.27068772e-02, 2.08120692e-01, 2.02835294e-01,
       7.19632891e-02, 4.92331624e-02, 1.69710288e-02, 4.44737076e-02,
       1.08191849e-01, 9.04933210e-02, 8.58326484e-02, 9.33790646e+12,
       9.33790646e+12, 9.33790646e+12])
```

[36]:
```
rfe = RFE(model,n_features_to_select=6)             # running RFE
rfe = rfe.fit(x_train, y_train)
```

[37]:
```
list(zip(x_train.columns,rfe.support_,rfe.ranking_))
```

[37]:
```
[('area', True, 1),
 ('bedrooms', False, 8),
 ('bathrooms', True, 1),
 ('stories', True, 1),
 ('parking', False, 2),
 ('mainroad', False, 4),
 ('guestroom', False, 9),
 ('basement', False, 5),
 ('hotwaterheating', True, 1),
 ('airconditioning', True, 1),
 ('prefarea', True, 1),
 ('furnished', False, 6),
 ('semi-furnished', False, 7),
 ('unfurnished', False, 3)]
```

Here it shows the features that are ordered based on the ranking and support of the dataset

[39]:
```
col = x_train.columns[rfe.support_]
print(col)
```

```
Index(['area', 'bathrooms', 'stories', 'hotwaterheating', 'airconditioning',
       'prefarea'],
      dtype='object')
```

[40]:
```
col = list(col)  # Convert the Index to a list
col.append('bedrooms')
col.pop(3)
```

[40]:
```
'hotwaterheating'
```

These columns are the important features in the dataset

[42]:
```
col
```

[42]:
```
['area', 'bathrooms', 'stories', 'airconditioning', 'prefarea', 'bedrooms']
```

[43]:
```
x_train_rfe = x_train[col]
x_train_rfe.head()
```

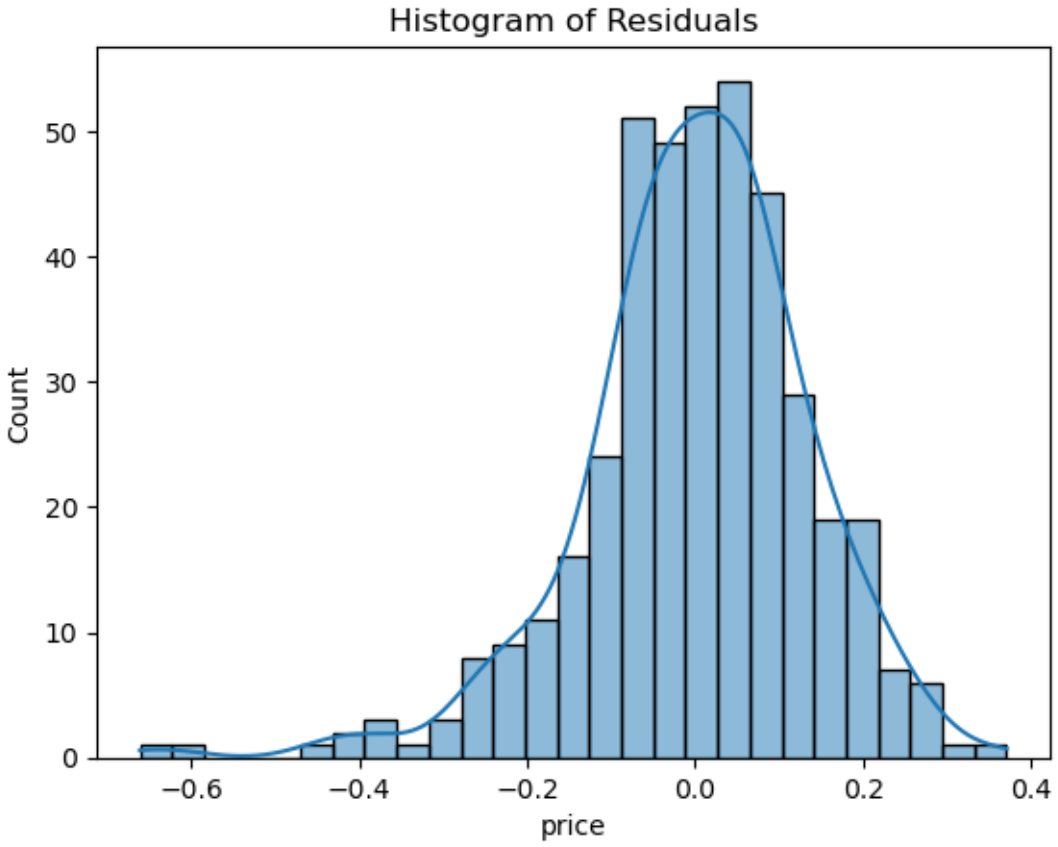
[43]:

	area	bathrooms	stories	airconditioning	prefarea	bedrooms
412	0.224018	0.0	0.333333	0	0	0.6
332	0.219400	0.0	0.000000	0	0	0.4
423	0.583141	0.0	0.000000	0	0	0.2
387	0.356236	0.0	0.333333	1	0	0.4
153	0.502309	0.0	0.000000	0	1	0.4

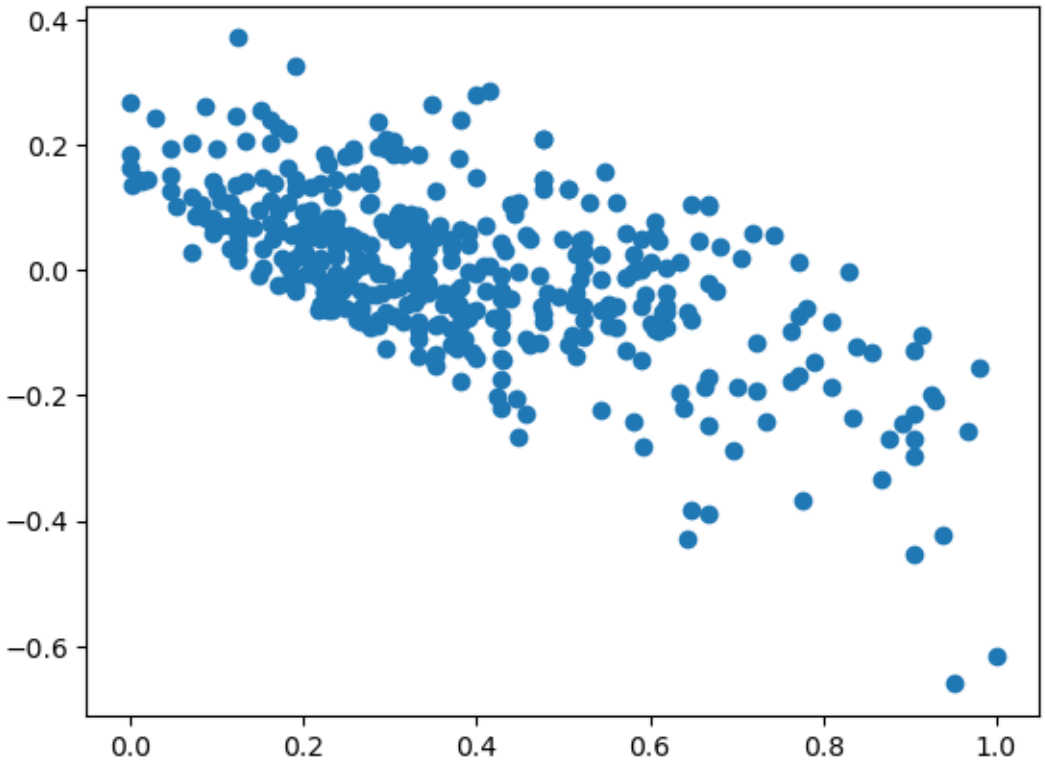
7 Residual Analysis

[45]:
```
model.fit(x_train_rfe,y_train)
y_pred_train = model.predict(x_train_rfe)
res = y_pred_train - y_train
```

[46]:
```
import seaborn as sns
sns.histplot(res, kde=True)
plt.title("Histogram of Residuals")
plt.show()
```



[47]:
```
plt.scatter(y_train,res)
plt.show()
```



8 Checking the loss values using cost function MSE

[49]:
```
from sklearn.metrics import mean_squared_error

mse_train = mean_squared_error(y_train, y_pred_train)
mse_train
```

[49]:
```
0.018076949225282565
```

[50]:
```
y_test = x_test.pop('price')
y_test.head()
```

[50]:
```
217    0.380952
260    0.333333
142    0.485714
152    0.476190
255    0.342857
Name: price, dtype: float64
```

[51]:
```
x_test_rfe = x_test[col]
x_test_rfe.head()
```

[51]:

	area	bathrooms	stories	airconditioning	prefarea	bedrooms
217	0.418014	0.0	0.333333	0	1	0.4
260	0.438799	0.0	0.333333	1	0	0.4
142	0.704388	0.0	0.000000	1	1	0.4
152	0.787529	0.0	0.000000	1	0	0.4
255	0.054850	0.0	0.333333	1	0	0.4

[52]:
```
# model = LinearRegression()
# model.fit(x_test_rfe,y_test)
y_pred_test = model.predict(x_test_rfe)
```

[53]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred_test)
```

[53]:
```
0.5565292492072347
```

[54]:
```
model.coef_
```

[54]:
```
array([0.41677226, 0.23614118, 0.18103804, 0.09384617, 0.10369782,
       0.06153342])
```

[55]:
```
mse_test = mean_squared_error(y_test, y_pred_test)
mse_test
```

[55]:
```
0.02340970898254077
```

[56]:
```
# Getting the cost funcion values of training and testing data

print(f"The cost function value of training data: {mse_train}")
print(f"The cost function value of testing data: {mse_test}")
```

```
The cost function value of training data: 0.018076949225282565
The cost function value of testing data: 0.02340970898254077
```

The model is well generalized over the dataset

[58]:
```
y_test.shape
y_test_matrix = y_test.values.reshape(-1,1)
y_test_matrix[:5]
```

[58]:
```
array([[0.38095238],
       [0.33333333],
       [0.48571429],
       [0.47619048],
       [0.34285714]])
```

[59]:
```
#load actual and predecited values side by side
dframe=pd.DataFrame({'actual':y_test_matrix.flatten(),'Predicted':y_pred_test.
 ↪flatten()})
#flatten toget single axis of data (1 dimension only)
```

[60]:
```
dframe.head(15)
```

[60]:

	actual	Predicted
0	0.380952	0.443430
1	0.333333	0.442241
2	0.485714	0.596283
3	0.476190	0.527236
4	0.342857	0.282222
5	0.785714	0.705067
6	0.238095	0.294030
7	0.360952	0.320482
8	0.419048	0.611487
9	0.571429	0.636487
10	0.190476	0.163110
11	0.219048	0.395060
12	0.238095	0.284681
13	0.185714	0.261580
14	0.161905	0.140989

[ ]:
```

```