Bengaluru House Price Machine Learning (Name - Akhil A , College - NBNSSOE)

August 21, 2020

1 Bengaluru House Price Machine Learning

1.1 Objective

To find a model that can predict the price of the model according to different factors.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
import warnings
```

[2]:
```
from sklearn.linear_model import LinearRegression,Lasso, ElasticNet, Ridge
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.externals import joblib
```

```
/home/akhil/anaconda3/lib/python3.7/site-
packages/sklearn/externals/joblib/__init__.py:15: FutureWarning:
sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23.
Please import this functionality directly from joblib, which can be installed
with: pip install joblib. If this warning is raised when loading pickled models,
you may need to re-serialize those models with scikit-learn 0.21+.
  warnings.warn(msg, category=FutureWarning)
```

1.1.1 DataSet

[3]:
```
df=pd.read_csv("Bengaluru/Bengaluru_House_Data.csv")
```

[4]:
```
warnings.filterwarnings('ignore')
```

[5]:
```
df.head()
```

[5]:
```
               area_type   availability                    location       size  \
0  Super built-up  Area          19-Dec  Electronic City Phase II      2 BHK
1            Plot  Area  Ready To Move          Chikka Tirupathi  4 Bedroom
2        Built-up  Area  Ready To Move               Uttarahalli      3 BHK
3  Super built-up  Area  Ready To Move        Lingadheeranahalli      3 BHK
4  Super built-up  Area  Ready To Move                  Kothanur      2 BHK

  society total_sqft  bath  balcony   price
0  Coomee        1056   2.0      1.0   39.07
1  Theanmp       2600   5.0      3.0  120.00
2      NaN       1440   2.0      3.0   62.00
3  Soiewre       1521   3.0      1.0   95.00
4      NaN       1200   2.0      1.0   51.00
```

[6]:
```
df.shape
```

[6]:
```
(13320, 9)
```

[7]:
```
df.isnull().sum()
```

[7]:
```
area_type          0
availability       0
location           1
size              16
society         5502
total_sqft         0
bath              73
balcony          609
price              0
dtype: int64
```

1.1.2 Cleaning Data

[8]:
```
df.bath.fillna(value=int(df.bath.mode()),inplace=True)
df.balcony.fillna(value=int(df.balcony.mode()),inplace=True)
```

[9]:
```
df.drop(["society"],axis=1,inplace=True)
```

[10]:
```
df[["location","size","balcony","bath"]].dropna(inplace=True)
```

[11]:
```
df.dropna(inplace=True)
```

[12]:
```
df.isna().sum()
```

[12]:

```
area_type       0
availability    0
location        0
size            0
total_sqft      0
bath            0
balcony         0
price           0
dtype: int64
```

[13]:

```
df["total_sqft"]=df["total_sqft"].str.extract('(\d+)',expand=True)
df["total_sqft"]=pd.to_numeric(df["total_sqft"])
```

[14]:

```
sz=df["size"]
df["room"]=sz.str.extract('([A-z]\w{0,})',expand=True)
df["no_room"]=sz.str.extract('(\d+)',expand=True)
df.drop("size",axis=1,inplace=True)
```

[15]:

```
df["no_room"]=pd.to_numeric(df["no_room"])
```

[16]:

```
df["room"][df.room=="Bedroom"]="BHK"
```

[17]:

```
df.availability[df.availability!="Ready To Move"]="Available Soon"
```

[18]:

```
for col in df.columns:
    print(col,df[col].unique())
```

```
area_type ['Super built-up  Area' 'Plot  Area' 'Built-up  Area' 'Carpet  Area']
availability ['Available Soon' 'Ready To Move']
location ['Electronic City Phase II' 'Chikka Tirupathi' 'Uttarahalli' …
 '12th cross srinivas nagar banshankari 3rd stage' 'Havanur extension'
 'Abshot Layout']
total_sqft [1056 2600 1440 … 2758  774 4689]
bath [ 2.  5.  3.  4.  6.  1.  9.  8.  7. 11. 10. 14. 27. 12. 16. 40. 15. 13.
 18.]
balcony [1. 3. 2. 0.]
price [ 39.07 120.    62.   …  40.14 231.   488.  ]
room ['BHK' 'RK']
no_room [ 2  4  3  6  1  8  7  5 11  9 27 10 19 16 43 14 12 13 18]
```

1.1.3 Outliner

[19]:

```
def total_sqft(df):
    tsm=df.total_sqft.mean()
    tsd=df.total_sqft.std()
    df=df[(df.total_sqft>tsm-tsd) & (df.total_sqft<tsm+tsd)]
    return df
```

[20]:

```
df=total_sqft(df)
```

[21]:

```
def locns(df):
    df_new=pd.DataFrame()
    for nm,dt in df.groupby("location"):
        m=dt["price"].mean()
        sdt=dt["price"].std()
        red_df=dt[(dt.price>m-sdt) & (dt.price<m+sdt)]
        df_new=pd.concat([df_new,red_df],ignore_index=True)
    return df_new
```

[22]:

```
df=locns(df)
```

[23]:

```
def locns2(df):
    df_new=pd.DataFrame()
    for nm,dt in df.groupby("location"):
        if dt.room.count()<15:
            continue
        else:
            df_new=pd.concat([df_new,dt],ignore_index=True)
    return df_new
```

[24]:

```
df=locns2(df)
```

[25]:

```
def price_adjusting(df):
    f_max=df.total_sqft.max()
    f_min=df.total_sqft.min()
    df_new=pd.DataFrame()
    for i in range(f_min,f_max,101):
        if i==f_max:
            break
        else:
            max_v=i+101
        da=df[(df.total_sqft>i) & (df.total_sqft<=max_v)]
        p_std=da.price.std()
        p_mean=da.price.mean()
        da=da[(da.price>p_mean-p_std) & (da.price<p_mean+p_std)]
        df_new=pd.concat([df_new,da])
    return df_new
```

[26]:

```
df=price_adjusting(df)
```

[27]:

```
for n in df.bath.unique():
    print(n,df.bath[df.bath==n].count())
```

```
1.0 262
```

```
2.0 3791
6.0 7
3.0 1373
4.0 149
5.0 20
7.0 4
8.0 2
```

[28]:

```
df=df[df.bath<6]
```

[29]:

```
df.drop_duplicates(inplace=True)
```

[30]:

```
for col in df.columns:
    print(col,df[col].unique(),df[col].dtypes)
```

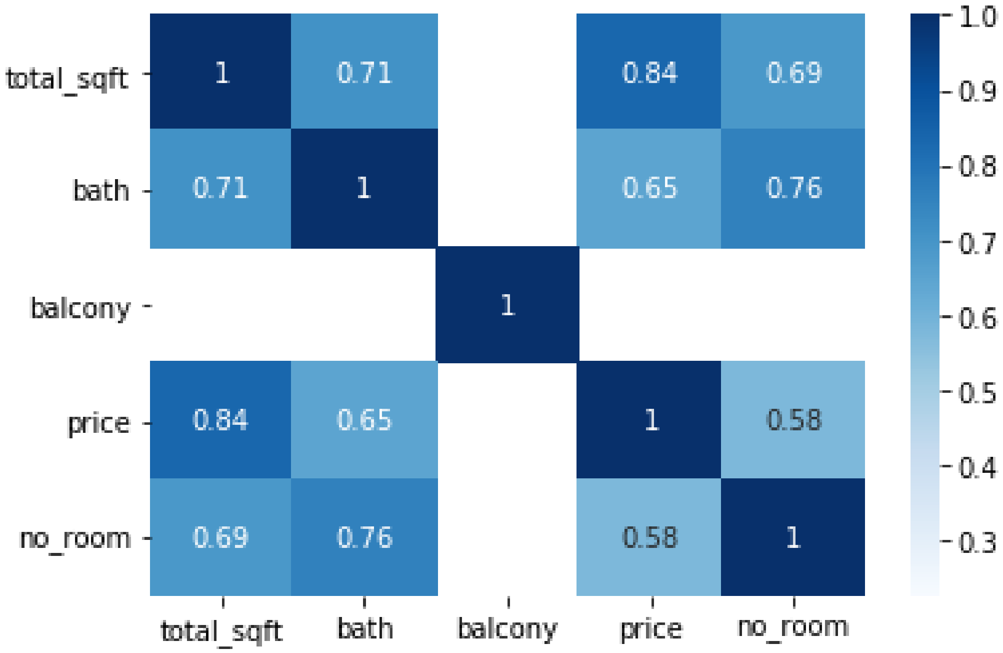
```
area_type ['Super built-up  Area' 'Built-up  Area' 'Carpet  Area' 'Plot  Area']
object
availability ['Ready To Move' 'Available Soon'] object
location ['Attibele' 'Electronic City' 'Kengeri' 'Yelahanka New Town'
 '8th Phase JP Nagar' 'Anekal' 'Bannerghatta Road'
 'Electronic City Phase II' 'Hoskote' 'Kogilu' 'Ramamurthy Nagar'
 'Sarjapur' 'Sarjapur  Road' 'Vijayanagar' 'Whitefield'
 '5th Phase JP Nagar' '6th Phase JP Nagar' '9th Phase JP Nagar'
 'Amruthahalli' 'Anandapura' 'Balagere' 'Banashankari' 'Begur Road'
 'Chandapura' 'Domlur' 'Electronics City Phase 1' 'Haralur Road'
 'Hosa Road' 'Jalahalli' 'Jigani' 'KR Puram' 'Kalena Agrahara'
 'Kengeri Satellite Town' 'Kodichikkanahalli' 'Kumaraswami Layout'
 'Nagarbhavi' 'Padmanabhanagar' 'Rachenahalli' 'Raja Rajeshwari Nagar'
 'Sanjay nagar' 'Singasandra' 'Sonnenahalli' 'Subramanyapura' 'TC Palaya'
 'Thanisandra' 'Vidyaranyapura' 'Yelahanka' 'Yeshwanthpur'
 '7th Phase JP Nagar' 'Akshaya Nagar' 'Bisuvanahalli' 'Bommanahalli'
 'CV Raman Nagar' 'Choodasandra' 'Horamavu Agara' 'Horamavu Banaswadi'
 'Hormavu' 'Hulimavu' 'JP Nagar' 'Kaggalipura' 'Kammasandra' 'Kanakapura'
 'Kanakpura Road' 'Kathriguppe' 'Kothannur' 'Kudlu Gate' 'Kundalahalli'
 'Marathahalli' 'Uttarahalli' 'Abbigere' 'Ananth Nagar' 'Bellandur'
 'Bhoganhalli' 'Bommasandra' 'Channasandra' 'Devanahalli' 'Dodda Nekkundi'
 'Doddathoguru' 'Hebbal' 'Hoodi' 'Indira Nagar' 'Kadugodi' 'Magadi Road'
 'Pai Layout' 'Ramagondanahalli' 'Seegehalli' 'BTM 2nd Stage'
 'Brookefield' 'Chikkalasandra' 'Gottigere' 'Hennur Road' 'Hosur Road'
 'Kambipura' 'Kaval Byrasandra' 'Margondanahalli' 'Munnekollal'
 'Mysore Road' 'Old Madras Road' 'Poorna Pragna Layout' 'Babusapalaya'
 'Bommasandra Industrial Area' 'HSR Layout' 'Hennur' 'Kaggadasapura'
 'Kudlu' 'OMBR Layout' 'Panathur' 'R.T. Nagar' 'Rayasandra'
 'Sahakara Nagar' 'Thubarahalli' 'Tumkur Road' 'Varthur' 'Ambalipura'
 'Banashankari Stage III' 'Budigere' 'Garudachar Palya' 'Gunjur' 'Jakkur'
 'Kasavanhalli' 'Kothanur' 'Mahadevpura' 'Parappana Agrahara'
 'Talaghattapura' '1st Phase JP Nagar' 'Ardendale' 'Basavangudi' 'Harlur'
 'Hegde Nagar' 'Lakshminarayana Pura' 'Somasundara Palya' 'Vittasandra'
 'EPIP Zone' 'Koramangala' 'Ulsoor' 'Green Glen Layout'
 'Lingadheeranahalli' 'Old Airport Road' 'Rajaji Nagar' 'Ambedkar Nagar'
 'Hebbal Kempapura' 'Thigalarapalya' 'Frazer Town' 'Malleshwaram'] object
total_sqft [ 450  400  395 … 2732 2774 2690] int64
bath [1. 2. 3. 4. 5.] float64
balcony [1. 0. 2. 3.] float64
price [ 11.  12.  14. … 201. 194. 197.] float64
room ['BHK'] object
no_room [1 2 3 4 5 8 7] int64
```

[31]:
```
df.shape
```

[31]:
```
(5159, 9)
```

[32]:
```
sns.heatmap(df.corr(),cmap="Blues",mask=df.corr()<0.5,annot=True)
```

[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0242d27c90>
```



[33]:
```
df.drop(columns=["area_type","availability"],inplace=True)
df.head()
```

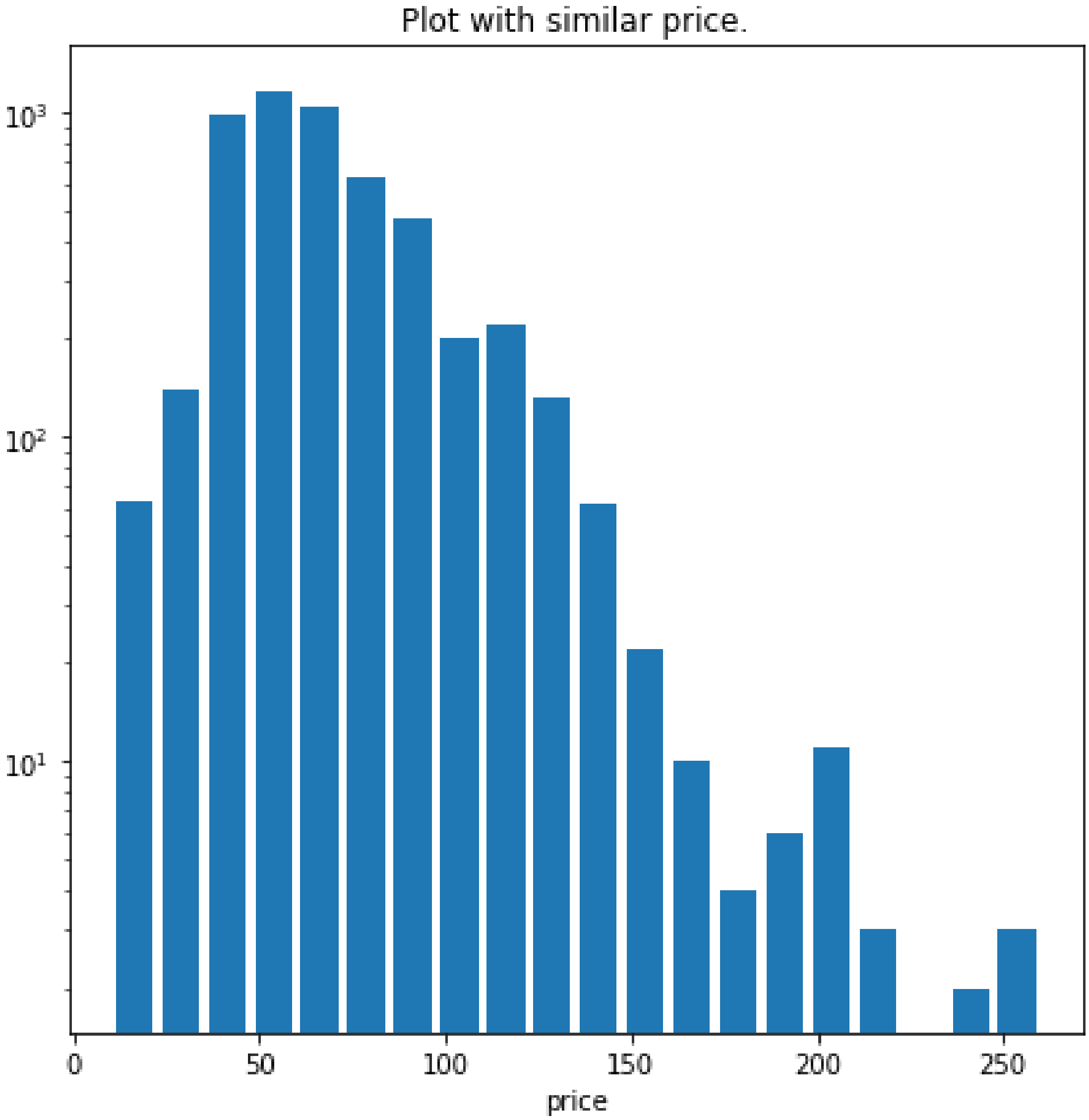
[33]:
```
      location  total_sqft  bath  balcony  price room  no_room
480  Attibele         450   1.0      1.0  11.00  BHK        1
483  Attibele         400   1.0      1.0  11.00  BHK        1
484  Attibele         400   1.0      1.0  12.00  BHK        1
485  Attibele         400   1.0      1.0  14.00  BHK        1
```

```
487  Attibele          395   1.0       1.0  10.25  BHK        1
```

1.1.4 Graphs

Number of plots within same price range.

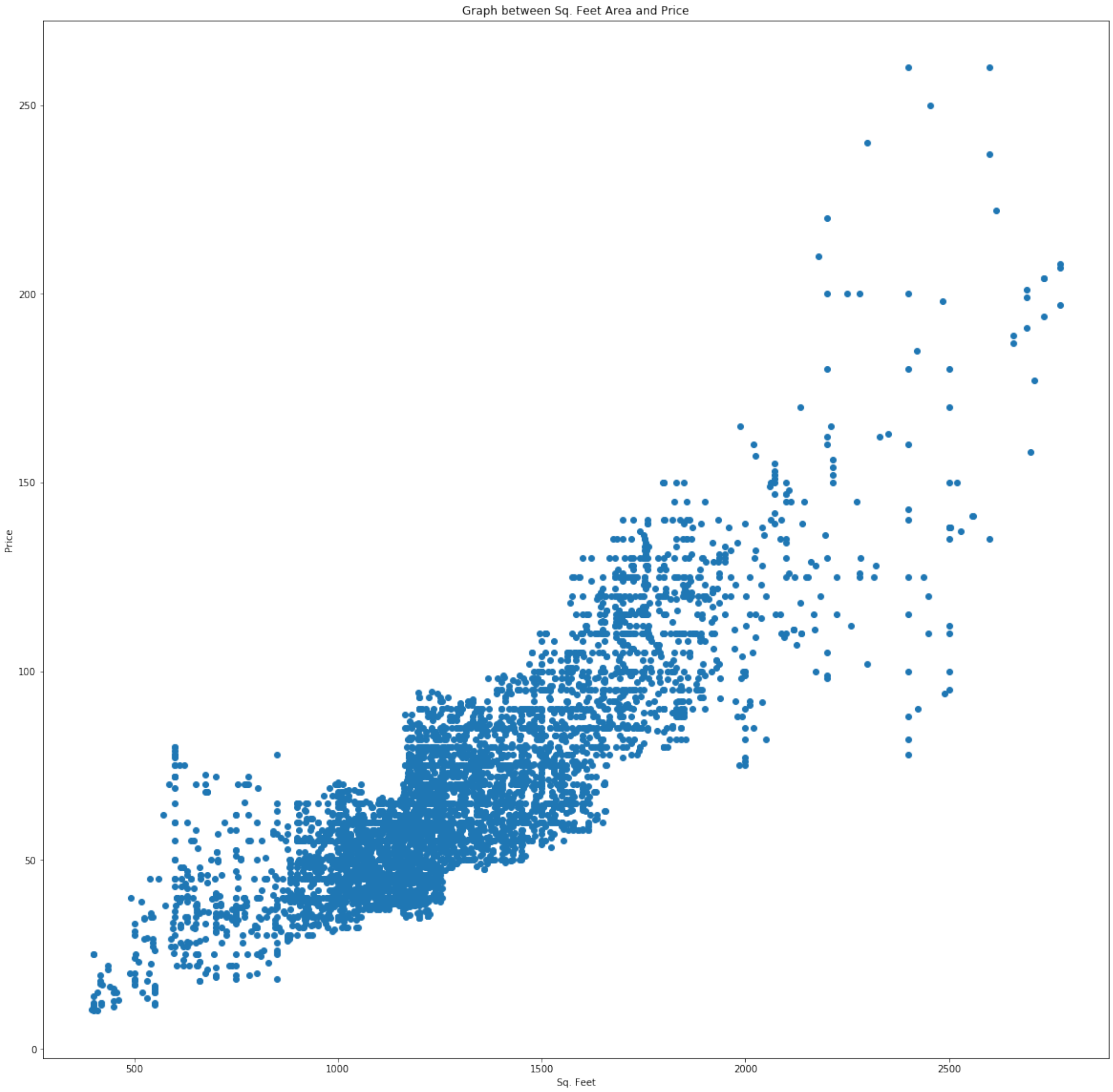
```
[34]: plt.figure(figsize=(7,7))
plt.hist(df.price,bins=20, rwidth=0.8)
plt.title("Plot with similar price.")
plt.xlabel("price")
plt.yscale("log")
```



Relation between the Area and Price of the Plot

[35]:

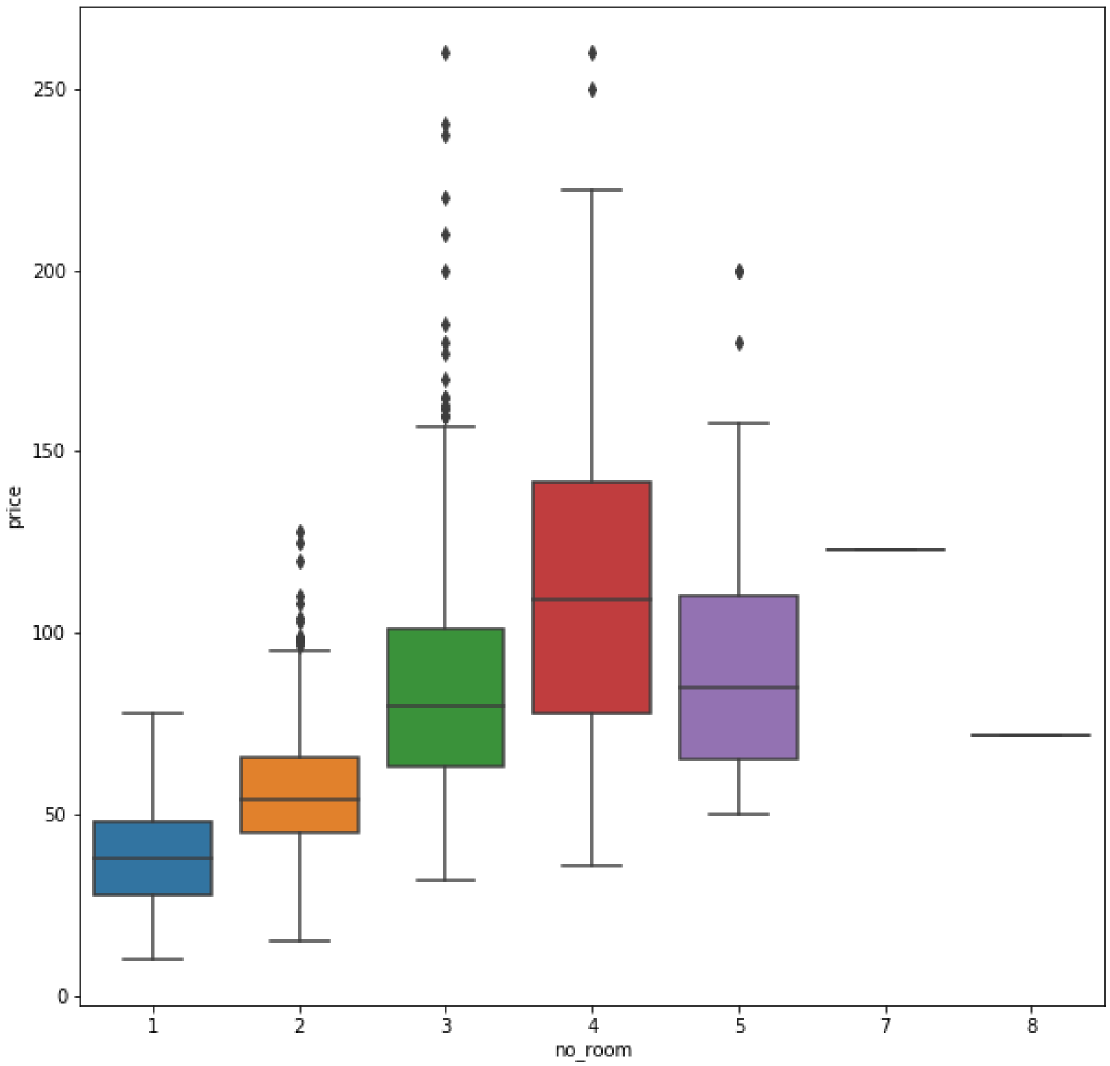
```
plt.figure(figsize=(20,20))
plt.title("Graph between Sq. Feet Area and Price")
plt.xlabel("Sq. Feet")
plt.ylabel("Price")
plt.scatter(df.total_sqft,df.price)
plt.show()
```



Plot of number of room and the respective price.

[36]:

```
plt.figure(figsize=(10,10))
sns.boxplot(df.no_room,df.price)
plt.show()
```



[37]:
```
df.index=(range(len(df)))
```

[38]:
```
df.index=(range(len(df)))
```

[39]:
```
lon=df.location.unique()
lon.sort()
```

1.1.5 Encoding

[40]:
```
le=LabelEncoder()
df.location=le.fit_transform(df.location)
df.room=le.fit_transform(df.room)
```

```
[41]: l=pd.DataFrame(df.location,columns=["location"])
      ct3=ColumnTransformer([("location",OneHotEncoder(),[0])],remainder="passthrough")
```

```
[42]: l=pd.DataFrame(ct3.fit_transform(l).todense(),columns=lon)
```

```
[43]: df=pd.concat([df,l],axis=1)
```

```
[44]: df.shape
```

```
[44]: (5159, 151)
```

```
[45]: df.drop("location",axis=1,inplace=True)
```

```
[46]: df.shape
```

```
[46]: (5159, 150)
```

```
[47]: df.head()
```

```
[47]:    total_sqft  bath  balcony  price  room  no_room  1st Phase JP Nagar  \
      0         450   1.0      1.0  11.00     0        1                 0.0
      1         400   1.0      1.0  11.00     0        1                 0.0
      2         400   1.0      1.0  12.00     0        1                 0.0
      3         400   1.0      1.0  14.00     0        1                 0.0
      4         395   1.0      1.0  10.25     0        1                 0.0

         5th Phase JP Nagar  6th Phase JP Nagar  7th Phase JP Nagar  …  Ulsoor  \
      0                 0.0                 0.0                 0.0  …     0.0
      1                 0.0                 0.0                 0.0  …     0.0
      2                 0.0                 0.0                 0.0  …     0.0
      3                 0.0                 0.0                 0.0  …     0.0
      4                 0.0                 0.0                 0.0  …     0.0

         Uttarahalli  Varthur  Vidyaranyapura  Vijayanagar  Vittasandra  Whitefield  \
      0          0.0      0.0             0.0          0.0          0.0         0.0
      1          0.0      0.0             0.0          0.0          0.0         0.0
      2          0.0      0.0             0.0          0.0          0.0         0.0
      3          0.0      0.0             0.0          0.0          0.0         0.0
      4          0.0      0.0             0.0          0.0          0.0         0.0

         Yelahanka  Yelahanka New Town  Yeshwanthpur
      0        0.0                 0.0           0.0
      1        0.0                 0.0           0.0
      2        0.0                 0.0           0.0
      3        0.0                 0.0           0.0
      4        0.0                 0.0           0.0
```

```
[5 rows x 150 columns]
```

[48]:

```
df.dropna(inplace=True)
```

[49]:

```
df.drop_duplicates(inplace=True)
```

[50]:

```
df.shape
```

[50]:

```
(5107, 150)
```

1.1.6 Training and Testing

[51]:

```
X=df.drop("price",axis=1)
y=df.price
```

[52]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X=sc.fit_transform(X)
```

[53]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

[54]:

```
Data={}
```

[55]:

```
lr=LinearRegression()
lr.fit(X_train,y_train)
lr.score(X_test,y_test)
Data["lr"]=lr.score(X_test,y_test)
```

[56]:

```
la=Lasso()
la.fit(X_train,y_train)
la.score(X_test,y_test)
Data["la"]=la.score(X_test,y_test)
```

[57]:

```
rfc=RandomForestRegressor()
rfc.fit(X_train,y_train)
rfc.score(X_test,y_test)
Data["rfc"]=rfc.score(X_test,y_test)
```

[58]:

```
dtr=DecisionTreeRegressor()
dtr.fit(X_train,y_train)
dtr.score(X_test,y_test)
Data["dtr"]=dtr.score(X_test,y_test)
```

[59]:

```
en=ElasticNet()
en.fit(X_train,y_train)
en.score(X_test,y_test)
```

```
Data["en"]=en.score(X_test,y_test)
```

[60]:

```
rid=Ridge()
rid.fit(X_train,y_train)
rid.score(X_test,y_test)
Data["rid"]=rid.score(X_test,y_test)
```

[61]:

```
svr=SVR()
svr.fit(X_train,y_train)
svr.score(X_test,y_test)
Data["svr"]=svr.score(X_test,y_test)
```

[62]:

```
Data
```

[62]:

```
{'lr': -1.1747179087179493e+24,
 'la': 0.7126457546151976,
 'rfc': 0.7852240913010364,
 'dtr': 0.682139902117145,
 'en': 0.6928666406090682,
 'rid': 0.751244297242834,
 'svr': 0.48614528677295243}
```

[63]:

```
clf=GridSearchCV(LinearRegression(),param_grid={},cv=3,return_train_score=False)
clf.fit(X_train,y_train)
clf.best_score_
```

[63]:

```
0.7867021355354084
```

[64]:

```
clf=GridSearchCV(Ridge(),param_grid={"alpha":[0.1,0.
 ↪5,1,2]},cv=3,return_train_score=False)
clf.fit(X_train,y_train)
clf.best_score_
```

[64]:

```
0.7867240745265316
```

[65]:

```
clf=GridSearchCV(Lasso(),param_grid={"alpha":[0.1,0.
 ↪5,1,2]},cv=3,return_train_score=False)
clf.fit(X_train,y_train)
clf.best_score_
```

[65]:

```
0.7876828683918088
```

[66]:

```
clf=GridSearchCV(SVR(),param_grid={"gamma":["auto","scaler"],"kernel":
 ↪["rbf","linear"]},cv=3,return_train_score=False)
clf.fit(X_train,y_train)
clf.best_score_
```

[66]:
```
0.7780137274957809
```

[67]:
```
clf=GridSearchCV(RandomForestRegressor(),param_grid={"n_estimators":
 ↪[1,10,20,50,100,200,300]},cv=3,return_train_score=False)
clf.fit(X_train,y_train)
print(clf.best_score_,clf.best_params_)
```

```
0.8009430264234668 {'n_estimators': 300}
```

1.2 Saving the best model for predicition

[68]:
```
clf=GridSearchCV(RandomForestRegressor(),param_grid={"n_estimators":
 ↪[300]},cv=3,return_train_score=False)
clf.fit(X_train,y_train)
clf.best_score_
```

[68]:
```
0.8009420516709219
```

[71]:
```
joblib.dump(clf,"Bengaluru_trained")
```

[71]:
```
['Bengaluru_trained']
```

[72]:
```
joblib.dump(y_test,"Bengaluru_test_result")
```

[72]:
```
['Bengaluru_test_result']
```

[73]:
```
joblib.dump(X_test,"Bengaluru_test")
```

[73]:
```
['Bengaluru_test']
```

Trying to load Data and Predict

[74]:
```
model=joblib.load("Bengaluru_trained")
test_result=joblib.load("Bengaluru_test_result")
test=joblib.load("Bengaluru_test")
```

[75]:
```
test=pd.DataFrame(test)
```

[76]:
```
model.score(test,test_result)
```

[76]:
```
0.7834556183550779
```