Project 1: Price Prediction Model of Washington DC

By – Yash Agrawal

This project includes 5 stages of implementation:

1. Problem Understanding:

The problem divides into 2 parts:

- To predict the price of the property based on the different features of the property such as No of rooms, area size, Sales data etc.
- To Get the effective factors that really affect the price of the property.

2. Data Understanding:

The data is in CSV format, It is a source of information on residential properties of Washington, DC including all the features and attributes connected to the property.

Used some data science libraries for the analysis and exploring the data.

1. Pandas 2. Numpy 3. Matplotlib

Below are the definition of each attributes include in the csv.

- BATHRM - Number of Full Bathrooms
- HF_BATHRM - Number of Half Bathrooms (no bathtub or shower)
- HEAT - Heating
- AC - Cooling
- NUM_UNITS - Number of Units
- ROOMS - Number of Rooms
- BEDRM - Number of Bedrooms
- AYB - The earliest time the main portion of the building was built
- YR_RMDL - Year structure was remodeled
- EYB - The year an improvement was built more recent than actual year built
- STORIES - Number of stories in primary dwelling
- SALEDATE - Date of most recent sale

- PRICE - Price of most recent sale
- QUALIFIED - Qualified
- SALE_NUM - Sale Number
- GBA - Gross building area in square feet
- BLDG_NUM - Building Number on Property
- STYLE - Style
- STRUCT - Structure
- GRADE - Grade
- CNDTN - Condition
- EXTWALL - Exterior wall
- ROOF - Roof type
- INTWALL - Interior wall
- KITCHEN - SNumber of kitchens
- FIREPLACES - Number of fireplaces
- LANDARE - ALand area of property in square feet
- WARD - Ward (District is divided into eight wards, each with approximately 75,000 residents).

This is a data attributes that will be used to make more predictions.

To load the data into Jupyter Notebook. I have applied Following Code:

data= pd.read_csv("DC_Properties_trimmed.csv")

Following data which I received after executing above command:

```
<bound method NDFrame.head of        BATHRM  HF_BATHRM          HEAT AC  NUM_UNITS  ROOMS  BEDRM   AYB  \
0           4          0      Warm Cool  Y          2      8      4  1910
1           3          1  Hot Water Rad  Y          2      9      5  1910
2           3          1  Hot Water Rad  Y          2      8      5  1900
3           3          1  Hot Water Rad  Y          2      8      4  1906
4           3          1      Warm Cool  Y          2      7      3  1908
5           3          1      Warm Cool  Y          2      5      3  1917
6           3          1      Warm Cool  Y          1      8      3  1908
7           3          1  Hot Water Rad  Y          2      9      3  1908
8           3          1  Hot Water Rad  Y          1     14      5  1880
9           1          0     Forced Air  Y          1      6      3  1880
10          2          1     Forced Air  Y          1      5      3  1880
11          2          1  Hot Water Rad  Y          1      8      3  1880
12          1          0  Hot Water Rad  N          1      8      4  1880
13          3          0  Hot Water Rad  Y          4      9      3  1900
14          3          1     Forced Air  Y          2     11      3  1900
15          2          1     Forced Air  Y          1      6      2  1890
16          2          2      Warm Cool  Y          2      8      4  1800
17          3          1     Forced Air  Y          2      9      4  1800
```

Understanding the Statistical and descriptive characteristics of the data:

```
data.describe()
```

	BATHRM	HF_BATHRM	NUM_UNITS	ROOMS	BEDRM	AYB	YR
count	28900.000000	28900.000000	28900.000000	28900.000000	28900.000000	28900.000000	28900
mean	2.333806	0.662007	1.261246	7.502872	3.482318	1922.556574	2004
std	1.038695	0.588201	0.635730	2.319767	1.160678	22.339850	17
min	0.000000	0.000000	0.000000	0.000000	0.000000	1754.000000	20
25%	2.000000	0.000000	1.000000	6.000000	3.000000	1908.000000	2002
50%	2.000000	1.000000	1.000000	7.000000	3.000000	1923.000000	2008
75%	3.000000	1.000000	1.000000	8.000000	4.000000	1938.000000	2012
max	11.000000	11.000000	6.000000	31.000000	20.000000	2015.000000	2018

3: Data preprocessing:

To check the presence of null value:

This is one of the most important step of analysis as this will reflect the performance of the models

To again Cross is there any missing value in entier data set

```
data.isnull().values.any()
```

```
False
```

This Result False shows that there is no missing value.

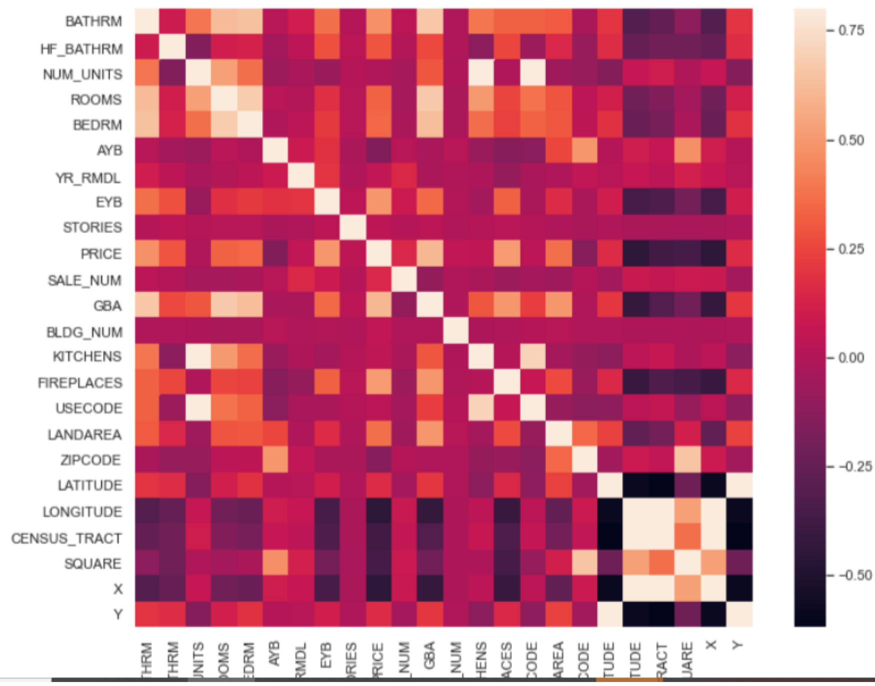
4. Exploratory Data Analysis:

For visualization I have used some libraries:

- Plotly
- Seaborn
- MatPlotlib

Plotting the heat map to check the correlation values among the data set varaible

```
corrmat = data.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
```



Finding the correlation Value among all the variables:

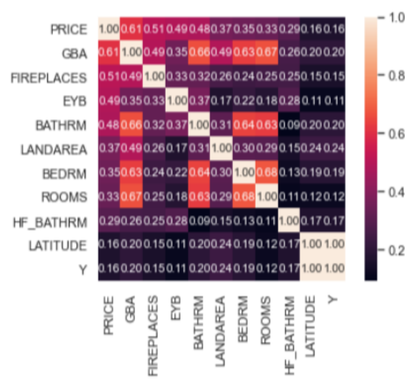
```
data.corr()
```

	BATHRM	HF_BATHRM	NUM_UNITS	ROOMS	BEDRM	AYB	YR_RMDL	EYB	STORIES	PRICE	...	FIREPLACES	USECODE
BATHRM	1.000000	0.091674	0.386041	0.627774	0.643595	0.019378	0.105670	0.369067	0.016594	0.478453	...	0.323764	0.328325
HF_BATHRM	0.091674	1.000000	-0.150110	0.107172	0.125967	-0.046838	0.036367	0.281119	0.030868	0.294055	...	0.252125	-0.068948
NUM_UNITS	0.386041	-0.150110	1.000000	0.525341	0.368530	-0.068746	-0.030503	-0.080477	0.007942	-0.000932	...	-0.006571	0.806691
ROOMS	0.627774	0.107172	0.525341	1.000000	0.679926	0.029107	0.002043	0.180071	0.019767	0.333632	...	0.250495	0.381186
BEDRM	0.643595	0.125967	0.368530	0.679926	1.000000	-0.000437	0.039898	0.217567	0.019235	0.347708	...	0.239889	0.330985
AYB	0.019378	-0.046838	-0.068746	0.029107	-0.000437	1.000000	0.087980	0.186923	-0.018659	-0.154585	...	-0.141191	-0.129635
YR_RMDL	0.105670	0.036367	-0.030503	0.002043	0.039898	0.087980	1.000000	0.192874	-0.002770	0.053331	...	-0.089801	-0.026330
EYB	0.369067	0.281119	-0.080477	0.180071	0.217567	0.186923	0.192874	1.000000	0.029507	0.490661	...	0.328570	-0.020103
STORIES	0.016594	0.030868	0.007942	0.019767	0.019235	-0.018659	-0.002770	0.029507	1.000000	0.043205	...	0.027550	0.008603
PRICE	0.478453	0.294055	-0.000932	0.333632	0.347708	-0.154585	0.053331	0.490661	0.043205	1.000000	...	0.510100	0.032633
SALE_NUM	0.033474	0.002992	-0.032901	-0.034648	-0.023032	0.022503	0.152992	0.090255	0.009658	0.147683	...	-0.069403	-0.044801
GBA	0.664008	0.259109	0.296131	0.667684	0.633087	-0.030361	-0.021864	0.352572	0.034900	0.607680	...	0.494267	0.225053
BLDG_NUM	-0.003781	-0.008241	-0.009462	-0.019034	-0.022629	0.019852	-0.003430	0.003206	0.000178	0.053984	...	-0.008569	0.012568
KITCHENS	0.392603	-0.119342	0.876726	0.501383	0.366895	-0.070786	-0.014051	-0.038331	0.009638	0.046747	...	0.014951	0.708746
FIREPLACES	0.323764	0.252125	-0.006571	0.250495	0.239889	-0.141191	-0.089801	0.328570	0.027550	0.510100	...	1.000000	0.065207
USECODE	0.328325	-0.068948	0.806691	0.381186	0.330985	-0.129635	-0.026330	-0.020103	0.008603	0.032633	...	0.065207	1.000000
LANDAREA	0.308770	0.150269	-0.058395	0.294609	0.303938	0.254659	-0.008213	0.166501	-0.008022	0.367954	...	0.264378	-0.082515
ZIPCODE	-0.015020	-0.086496	-0.085246	0.038441	0.043287	0.491707	0.054620	-0.022973	-0.018634	-0.138481	...	-0.080434	-0.118163
LATITUDE	0.195614	0.168548	-0.146765	0.116655	0.185634	0.015724	0.023048	0.110033	-0.000550	0.163768	...	0.154144	-0.112626
LONGITUDE	-0.315836	-0.247136	0.064331	-0.212389	-0.234877	0.098753	0.067552	-0.355761	-0.021140	-0.448443	...	-0.408666	0.034514
CENSUS_TRACT	-0.258171	-0.205561	0.101117	-0.156929	-0.189380	0.066991	0.039568	-0.328452	-0.017783	-0.372004	...	-0.329646	0.062152
SQUARE	-0.120040	-0.206323	-0.004543	-0.040251	-0.025397	0.470642	0.121353	-0.199514	-0.028011	-0.362841	...	-0.359215	-0.083375
X	-0.315940	-0.247075	0.064271	-0.212386	-0.234943	0.098548	0.067449	-0.355725	-0.021235	-0.448424	...	-0.408818	0.034478

f804fff0128868a6d148

Try to get top 10 attributes that are more correlated with predictive variable that is price:

```
k = 11
cols = corrmat.nlargest(k, 'PRICE')['PRICE'].index
cm = np.corrcoef(data[cols].values.T)
sns.set(font_scale=1.00)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10}, yticklabels=cols.values, xticklabels
plt.show()
```



Got the list of top 10 most correlated variables and preparing an dataframe including all the most effective variables:

```
To get the all top 10 variables that are used for correlation to predict the price of the property
```

```
most_corr = pd.DataFrame(cols)
most_corr.columns = ['Most Correlated Features']
most_corr
```

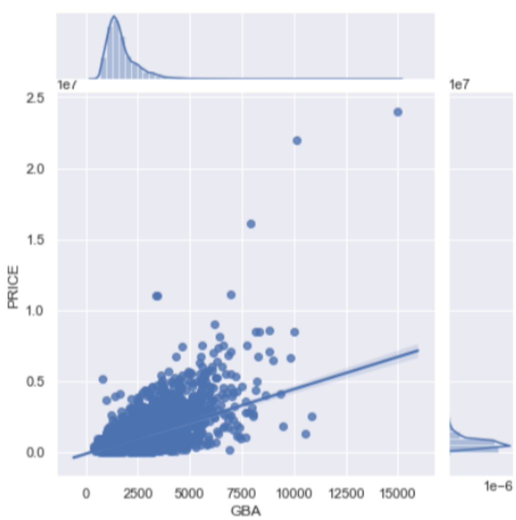
	Most Correlated Features
0	PRICE
1	GBA
2	FIREPLACES
3	EYB
4	BATHRM
5	LANDAREA
6	BEDRM
7	ROOMS
8	HF_BATHRM
9	LATITUDE
10	Y

```
data_select=data[['BATHRM', 'HF_BATHRM', 'ROOMS', 'BEDRM',
     'EYB', 'PRICE','LATITUDE',
       'GBA',  'FIREPLACES','LANDAREA', 'Y']]
```

Try to find hidden Pattern among the variables with predictive variable:

```
sns.jointplot(x=data_select['GBA'], y=data_select['PRICE'], kind='reg')
```

```
<seaborn.axisgrid.JointGrid at 0x2365e655518>
```




```
GBA means Gross building Area
This graph shows that with increasing the building area the price of the place will  increase gradually
```

To differentiate categorical and numerical variables

```
cat = len(data_select.select_dtypes(include=['object']).columns)
num = len(data_select.select_dtypes(include=['int64','float64']).columns)
```

To get total number of features

```
print('Total Features: ', cat, 'categorical', '+',
      num, 'numerical', '=', cat+num, 'features')
```

```
Total Features:  0 categorical + 11 numerical = 11 features
```

Here I got total 11 features having all the attributes of numerical characteristics

Exploring all the attributes with predictive attribute

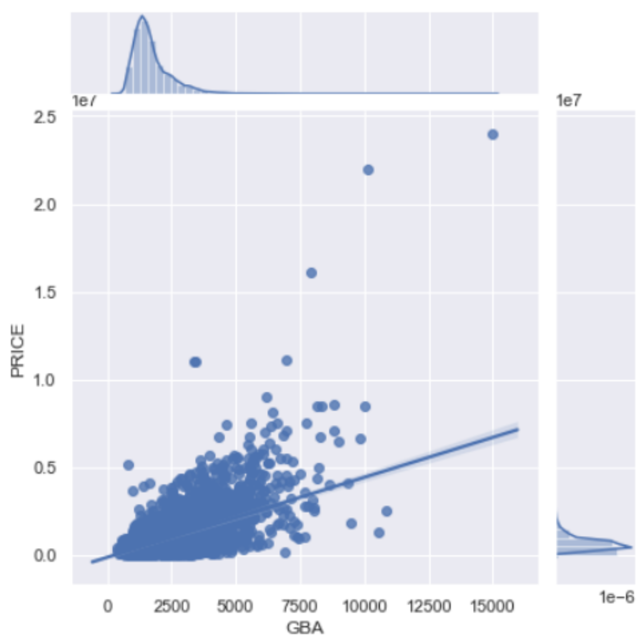
GROSS BUILDING AREA VS PRICE

Checking with GBA -Gross Building Area and price

To check the dependency between them

```
sns.jointplot(x=data_select['GBA'], y=data_select['PRICE'], kind='reg')
```

```
<seaborn.axisgrid.JointGrid at 0x2365e52b630>
```

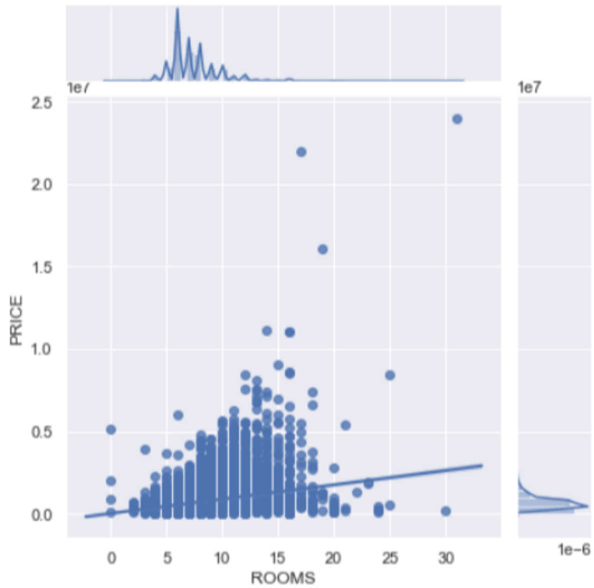


Through This It is understandable that People can pay more for more living area

ROOMS VS PRICE

```
sns.jointplot(x=data_select['ROOMS'], y=data_select['PRICE'], kind='reg')
```

```
<seaborn.axisgrid.JointGrid at 0x2365e968208>
```



Understood that maximum price lays for count of room lies between 10 to 15

Installing all the cufflinks and plotly for the different graphs using conda

```
#conda install -c conda-forge cufflinks-py
```

```
from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
```

```
print(__version__)
```

```
4.8.1
```

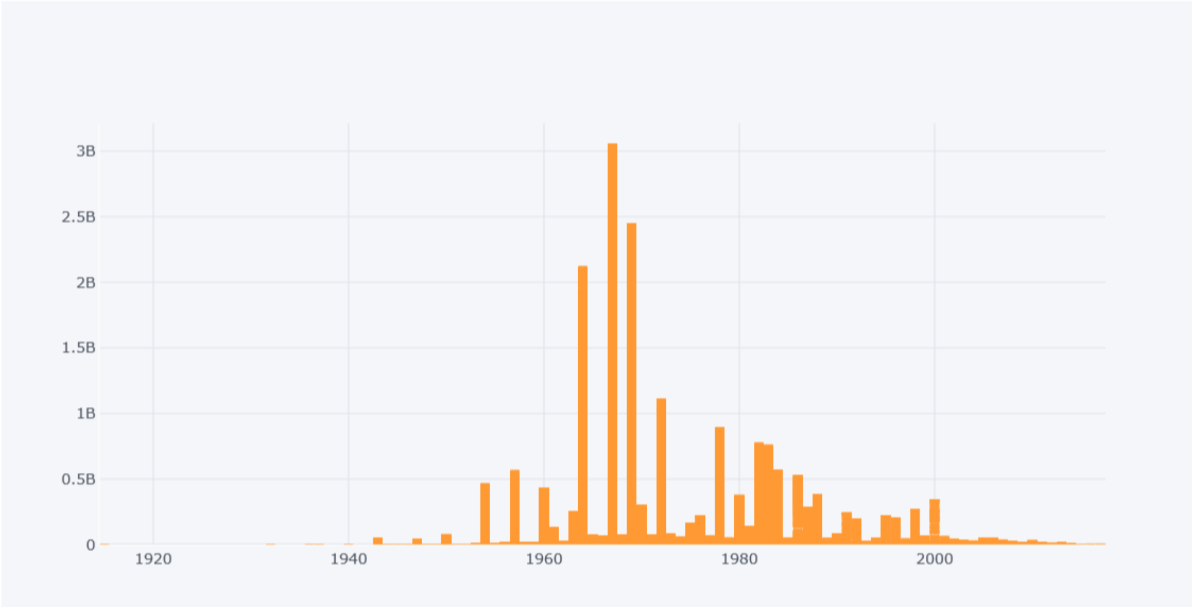
```
import cufflinks as cf
```

Used Cufflinks for connecting Plotly to jupyter notebook for better dynamic and interactive Graph.

Visualizing during among variables

Scatter Plot

```
data_select.iplot(kind = 'bar', x = 'EYB', y = 'PRICE')
```



EYB - The year an improvement was built more recent than actual year built

price of property is higher in 1967 , This shows that the property which improved in 1967 are more expensive, which is 360K.

5. Data models:

For applying Models on the pre-processed data we need to import some packages:

```
from scipy import stats as st
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import RFE, f_regression
from sklearn.ensemble import RandomForestRegressor
from scipy import stats
from scipy.stats import norm, skew #for some statistics
```

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
```

```
import statsmodels.formula.api as sm
```

```
# Splitting into train and test data
from sklearn.model_selection import train_test_split

train,test = train_test_split(data_select, train_size=0.8 , random_state=100)
```

After import we need to split the data into train and test.

This splitting is necessary because we train and build model on train data and check performance on the basis of test data.

Performance Measure for each model is R2, this is statistical measure that shows the proportion of variance for a dependent variable in our case (PRICE) that can be explain by the independent variables (Remaining all other attributes)

Model 1: OLS Model

```
import statsmodels.api as sm

from sklearn.metrics import r2_score

from sklearn.metrics import mean_squared_error

train_dataset_Y = data_select.PRICE.values

train_dataset_X = data_select.drop('PRICE',axis =1)

train_dataset_X.shape

train_dataset_X = sm.add_constant(train_dataset_X)

Pricing_model = sm.OLS(train_dataset_Y,train_dataset_X)

result = Pricing_model.fit()

print(result.summary())
```

Output from OLS regression model:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.512
Model:                            OLS   Adj. R-squared:                  0.512
Method:                 Least Squares   F-statistic:                     3028.
Date:                Fri, 26 Jun 2020   Prob (F-statistic):               0.00
Time:                        20:04:21   Log-Likelihood:            -4.1538e+05
No. Observations:               28900   AIC:                         8.308e+05
Df Residuals:                   28889   BIC:                         8.309e+05
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -2.01e+07   3.41e+06     -5.897      0.000   -2.68e+07   -1.34e+07
BATHRM      8.081e+04   3688.972     21.905      0.000    7.36e+04     8.8e+04
HF_BATHRM   7.595e+04   4597.019     16.523      0.000    6.69e+04     8.5e+04
ROOMS      -2.377e+04   1663.987    -14.282      0.000    -2.7e+04   -2.05e+04
BEDRM      -2.131e+04   3243.886     -6.570      0.000   -2.77e+04    -1.5e+04
EYB         1.363e+04    270.426     50.385      0.000    1.31e+04    1.42e+04
LATITUDE    1.424e+07   6.06e+06      2.351      0.019    2.37e+06    2.61e+07
GBA          268.9170      5.530     48.627      0.000     258.078     279.756
FIREPLACES  1.245e+05   3010.059     41.364      0.000    1.19e+05     1.3e+05
LANDAREA      18.2161      1.010     18.030      0.000      16.236      20.196
Y          -1.441e+07   6.06e+06     -2.379      0.017   -2.63e+07   -2.54e+06
==============================================================================
Omnibus:                    40388.076   Durbin-Watson:                   1.555
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         62901110.380
Skew:                           7.541   Prob(JB):                         0.00
Kurtosis:                     231.054   Cond. No.                     1.59e+07
==============================================================================
```

Here the output shows many parameter to check performance of the model:

Though using Performance metric R2 for this time, **For OLS the R2 is 0.52**

Ridge Model and LAsso Model

```
from sklearn.linear_model import Ridge

from sklearn.linear_model import RidgeCV

from sklearn.metrics import r2_score

## training the model

train_dataset_X = train_dataset_X.drop('const',axis=1)

regr_cv = RidgeCV(alphas=[0.1,1,2,3,4,5,6,7,0.5,0.8])

model_cv = regr_cv.fit(train_dataset_X,train_dataset_Y)
```

```
7.0
```

```
ridgeReg = Ridge(alpha=5, normalize=True)

ridgeReg.fit(train_dataset_X,train_dataset_Y)

pred = ridgeReg.predict(test_dataset)

# calculating mse

mse = np.sqrt(mean_squared_error(pred , test_dataset_Y))


print("The R2 value of Ridge Regression is ",r2_score(test_dataset_Y,pred))
```

```
The R2 value of Ridge Regression is  0.26492394270208586
```

```
from sklearn.linear_model import Lasso

lassoReg = Lasso(alpha=20, normalize=True)

lassoReg.fit(train_dataset_X,train_dataset_Y)

pred = lassoReg.predict(test_dataset)

# calculating mse

rmse = np.sqrt(mean_squared_error(pred, test_dataset_Y))

print("The R2 value of Lasso Regression is ",r2_score(test_dataset_Y,pred))
```

```
The R2 value of Lasso Regression is  0.4631844208406142
```

The R2 value for ridge is 0.26

The R2 value for Lasso is 0.46

Decision Tree model

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(train_dataset_X,train_dataset_Y)

# Predicting a new result
y_pred = regressor.predict(test_dataset)

rmse = np.sqrt(mean_squared_error(y_pred, test_dataset_Y))



print("The R2 value of Decision Tree Regression is ",r2_score(test_dataset_Y,y_pred))
```

```
The R2 value of Decision Tree Regression is  0.9999212045189719
```

K nearest Neighbour Algorithm

```
from sklearn import neighbors


knn = neighbors.KNeighborsRegressor(5)

pred_test = knn.fit(train_dataset_X,train_dataset_Y).predict(test_dataset)

RMSE = np.sqrt(mean_squared_error(test_dataset_Y, pred_test))


print("The R2 value of KNN Regression is ",r2_score(test_dataset_Y,pred_test))
```

```
The R2 value of KNN Regression is  0.5309968203649839
```

Random Forest Regression Algorithm

```
from sklearn.ensemble import RandomForestRegressor
# Instantiate model with 1000 decision trees
rf = RandomForestRegressor(n_estimators = 25)
# Train the model on training data
pred = rf.fit(train_dataset_X,train_dataset_Y).predict(test_dataset)

RMSE_1 = np.sqrt(mean_squared_error(test_dataset_Y, pred))
print("The R2 value of Random Forest Regression is ",r2_score(test_dataset_Y,pred))
```

```
The R2 value of Random Forest Regression is  0.9317517392653019
```

Gradient Boosting Regression Algorithm

```
from sklearn.ensemble import GradientBoostingRegressor

pred = GradientBoostingRegressor(n_estimators=100, learning_rate=0.3,max_depth=1, random_sta

RMSE_1 = np.sqrt(mean_squared_error(test_dataset_Y, pred))


print("The R2 value of Gradient Boosting Regression is ",r2_score(test_dataset_Y,pred))
```

```
The R2 value of Gradient Boosting Regression is  0.6444609605554794
```

6. Comparing Results:

Model Name	R2
1. OLS	0.52
2. Ridge	0.26
3. Lasso	0.46
4. Decision Tree	0.999
5. KNN	0.533
6. Random Forest	0.933
7. Gradient Descent	0.644

Conclusion:

As the value of R2 represent the dependency of the Price of the factor, so better the R2 value Better is the model to apply more data. In my analysis Decision Tree and Random Forest are the best models.

2. No of Bedrooms, Year of rebuilding the House, Gross base area are the most important factors for determining the price