In [189]:

```
# House Price Prediction using Machine Learning
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
```

In [132]:

```
# Importing data
df=pd.read_csv("C:/Users/ISHITA GUPTA/Desktop//data.csv")
```

In [133]:

```
df.head()
```

Out[133]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	vie
0	2014-05-02 00:00:00	313000.00000	3.00000	1.50000	1340	7912	1.50000	0	
1	2014-05-02 00:00:00	2384000.00000	5.00000	2.50000	3650	9050	2.00000	0	
2	2014-05-02 00:00:00	342000.00000	3.00000	2.00000	1930	11947	1.00000	0	
3	2014-05-02 00:00:00	420000.00000	3.00000	2.25000	2000	8030	1.00000	0	
4	2014-05-02 00:00:00	550000.00000	4.00000	2.50000	1940	10500	1.00000	0	

In [134]:

```
df.tail()
```

Out[134]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
4595	2014-07-09 00:00:00	308166.66667	3.00000	1.75000	1510	6360	1.00000	0
4596	2014-07-09 00:00:00	534333.33333	3.00000	2.50000	1460	7573	2.00000	0
4597	2014-07-09 00:00:00	416904.16667	3.00000	2.50000	3010	7014	2.00000	0
4598	2014-07-10 00:00:00	203400.00000	4.00000	2.00000	2090	6630	1.00000	0
4599	2014-07-10 00:00:00	220600.00000	3.00000	2.50000	1490	8102	2.00000	0

In [135]:

```
df.describe()
```

Out[135]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	water
count	4600.00000	4600.00000	4600.00000	4600.00000	4600.00000	4600.00000	4600.0
mean	551962.98847	3.40087	2.16082	2139.34696	14852.51609	1.51207	0.0
std	563834.70255	0.90885	0.78378	963.20692	35884.43614	0.53829	0.0
min	0.00000	0.00000	0.00000	370.00000	638.00000	1.00000	0.0
25%	322875.00000	3.00000	1.75000	1460.00000	5000.75000	1.00000	0.0
50%	460943.46154	3.00000	2.25000	1980.00000	7683.00000	1.50000	0.0
75%	654962.50000	4.00000	2.50000	2620.00000	11001.25000	2.00000	0.0
max	26590000.00000	9.00000	8.00000	13540.00000	1074218.00000	3.50000	1.0

In [136]:

```
df.columns
```

Out[136]:

```
Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'floors', 'waterfront', 'view', 'condition', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'street', 'city',
       'statezip', 'country'],
      dtype='object')
```

In [137]:

```
df.shape
```

Out[137]:

```
(4600, 18)
```

In [138]:

```
pd.set_option('display.float_format', lambda x: '%.5f' % x)
```

In [139]:

```
df.describe()
```

Out[139]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	water
count	4600.00000	4600.00000	4600.00000	4600.00000	4600.00000	4600.00000	4600.0
mean	551962.98847	3.40087	2.16082	2139.34696	14852.51609	1.51207	0.0
std	563834.70255	0.90885	0.78378	963.20692	35884.43614	0.53829	0.0
min	0.00000	0.00000	0.00000	370.00000	638.00000	1.00000	0.0
25%	322875.00000	3.00000	1.75000	1460.00000	5000.75000	1.00000	0.0
50%	460943.46154	3.00000	2.25000	1980.00000	7683.00000	1.50000	0.0
75%	654962.50000	4.00000	2.50000	2620.00000	11001.25000	2.00000	0.0
max	26590000.00000	9.00000	8.00000	13540.00000	1074218.00000	3.50000	1.0

In [140]:

```
# Distinct name of cities and their count
df['city'].value_counts()
```

Out[140]:

```
Seattle              1573
Renton                293
Bellevue              286
Redmond               235
Issaquah              187
Kirkland              187
Kent                  185
Auburn                176
Sammamish             175
Federal Way           148
Shoreline             123
Woodinville           115
Maple Valley           96
Mercer Island          86
Burien                 74
Snoqualmie             71
Kenmore                66
Des Moines             58
```

In [141]:

```
#check null values
df.isnull().sum()
```

Out[141]:

```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

In [ ]:

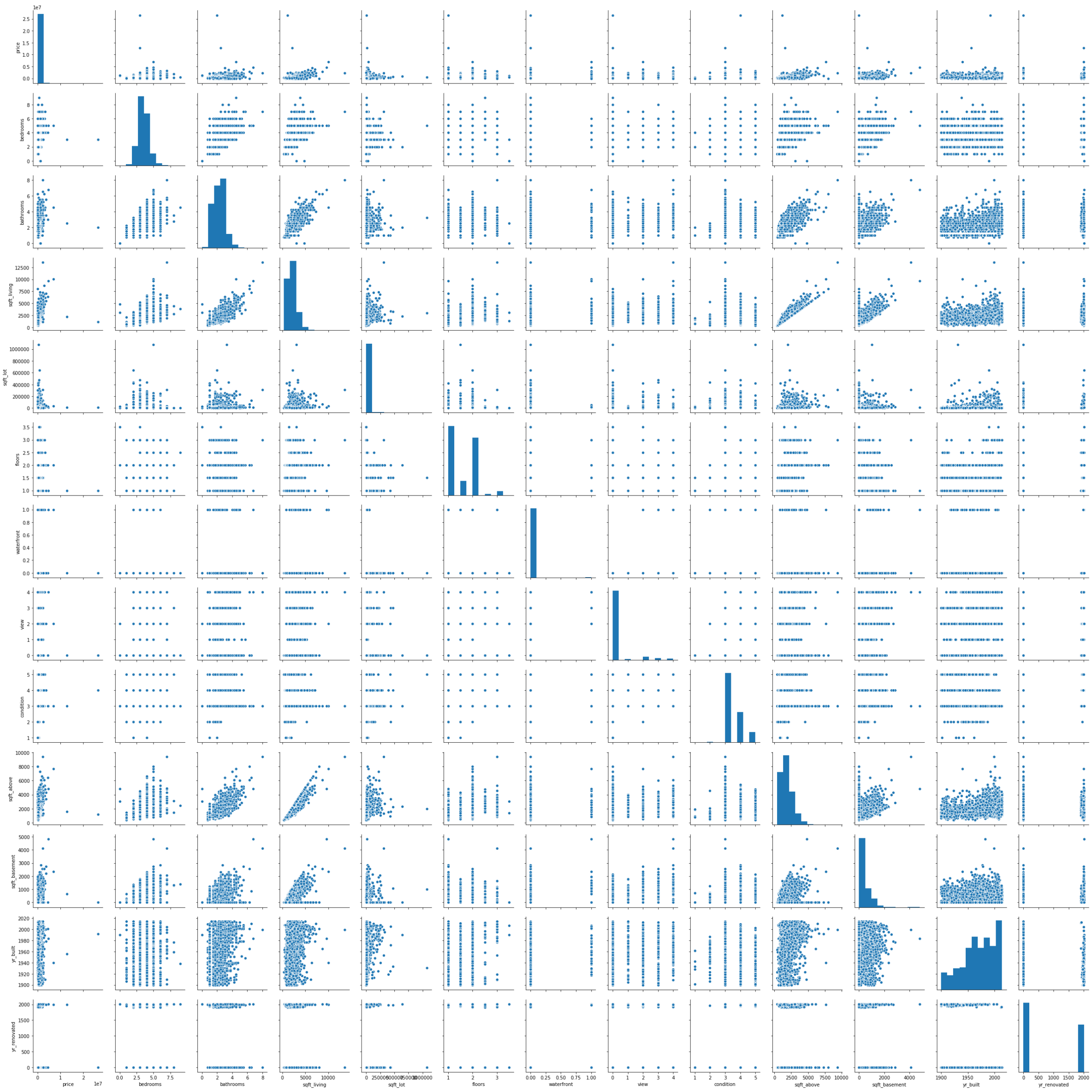
```
# Exploratory Data Analysis
```

In [142]:

```
#Plotting Data
sns.pairplot(df)
```

Out[142]:

```
<seaborn.axisgrid.PairGrid at 0x25373eeeef0>
```

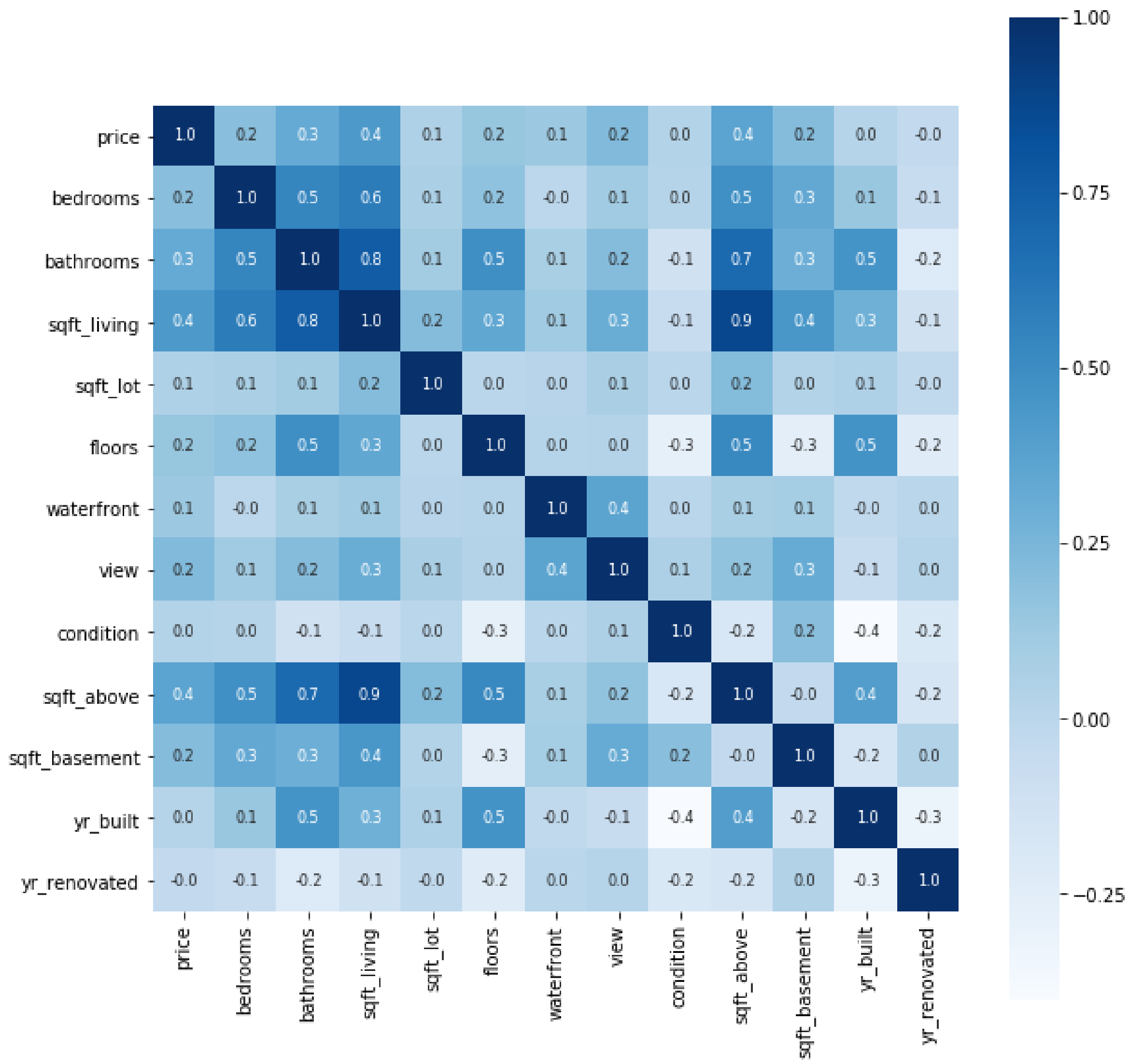


In [143]:

```
# constructing a heatmap to understand the correlation
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(), cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':8},
```

Out[143]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2537bddf358>
```



In [144]:

```
# Splitting data for training and testing
X = df[['sqft_living','condition','sqft_above','yr_built','floors']]
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

In [145]:

```
X_train
```

Out[145]:

	sqft_living	condition	sqft_above	yr_built	floors
695	3200	3	3200	2004	2.00000
1170	2350	4	1430	1977	1.00000
684	700	4	700	1949	1.00000
2490	1150	3	1150	1950	1.00000
2882	1290	4	1290	1906	2.00000
979	2850	4	1450	1970	1.00000
2601	1960	4	1960	1967	1.00000
4365	1600	3	1600	2013	2.00000
3627	3500	3	3500	2005	2.00000
19	1180	3	1180	1983	1.00000

In [146]:

```
X_test
```

Out[146]:

	sqft_living	condition	sqft_above	yr_built	floors
4032	5430	4	5430	1987	2.00000
1558	2270	3	1380	1977	1.00000
2004	2070	3	2070	2004	2.00000
3186	2550	5	1860	1902	2.00000
4176	1460	3	1460	1952	1.00000
3643	930	2	930	1942	1.00000
1970	2340	5	1170	1917	1.00000
2433	1070	3	1070	1999	2.00000
1407	1930	4	1030	1967	1.00000
2042	2030	3	1720	2001	2.00000

In [147]:

```
y_train
```

Out[147]:

```
695      707000.00000
1170     555000.00000
684      267800.00000
2490     129000.00000
2882     549000.00000
979     1275000.00000
2601     285500.00000
4365     444845.00000
3627     780000.00000
19       275000.00000
1168     150000.00000
469      842500.00000
1365     292050.00000
3756     600000.00000
1909     418500.00000
3626     835000.00000
4205     410000.00000
2439     589000.00000
```

In [148]:

```
y_test
```

```
1694     809000.00000
1292     606000.00000
1439     840000.00000
2685     493000.00000
524     1300000.00000
355      530000.00000
2919     245000.00000
2466     380000.00000
699      450000.00000
178      379900.00000
346      295000.00000
2775     175000.00000
1483     380000.00000
2725    1325000.00000
1544     672000.00000
3143     715000.00000
1105     220000.00000
3797     535000.00000
166      425000.00000
154      609000.00000
```

In [149]:

```
# Standardization
std=StandardScaler()
```

In [150]:

```
# Normalization
X_train_std=std.fit_transform(X_train)
X_test_std=std.transform(X_test)
```

In [151]:

```
X_train
```

Out[151]:

	sqft_living	condition	sqft_above	yr_built	floors
695	3200	3	3200	2004	2.00000
1170	2350	4	1430	1977	1.00000
684	700	4	700	1949	1.00000
2490	1150	3	1150	1950	1.00000
2882	1290	4	1290	1906	2.00000
979	2850	4	1450	1970	1.00000
2601	1960	4	1960	1967	1.00000
4365	1600	3	1600	2013	2.00000
3627	3500	3	3500	2005	2.00000
19	1180	3	1180	1983	1.00000

In [152]:

```
X_train_std
```

Out[152]:

```
array([[ 1.0980722 , -0.66150685,  1.60694763,  1.12296229,  0.88391385],
       [ 0.21123019,  0.80496616, -0.47145962,  0.21714364, -0.96287057],
       [-1.51028665,  0.80496616, -1.32865582, -0.72222384, -0.96287057],
       ...,
       [ 0.31556455,  0.80496616,  0.72626659,  0.25069248, -0.96287057],
       [-0.17272023, -0.66150685,  0.17672163,  1.12296229,  0.88391385],
       [ 0.31556455,  0.80496616, -0.69456548, -2.36611693, -0.96287057]])
```

In [153]:

```
X_test_std
```

Out[153]:

```
array([[ 3.42472829e+00,  8.04966165e-01,  4.22550590e+00,
         5.52632029e-01,  8.83913845e-01],
       [ 1.27762709e-01, -6.61506848e-01, -5.30171685e-01,
         2.17143643e-01, -9.62870570e-01],
       [-8.09059991e-02, -6.61506848e-01,  2.80054867e-01,
         1.12296229e+00,  8.83913845e-01],
       ...,
       [-6.54744945e-01,  2.27143918e+00, -3.65777892e-01,
        -1.18344744e-01, -9.62870570e-01],
       [ 2.56148399e-03, -6.61506848e-01,  3.73994178e-01,
         4.85534352e-01,  8.83913845e-01],
       [ 8.68536621e-01, -6.61506848e-01,  1.34861452e+00,
         9.88766931e-01,  8.83913845e-01]])
```

In [154]:

```
y_train
```

```
1273     440000.00000
3912     815000.00000
1580     672500.00000
2107     378500.00000
2931     480000.00000
1949     540000.00000
4467    1337044.20000
2184     740000.00000
1530     535000.00000
49       838000.00000
4573     584000.00000
908      110700.00000
3182     625000.00000
3829     599950.00000

2623     325000.00000
973      385000.00000
4079     513000.00000
4171     749950.00000
599      450000.00000
```

In [111]:

```
y_test
```

Out[111]:

```
4032    1360000.00000
1558     332000.00000
2004     343000.00000
3186     660000.00000
4176     310000.00000
3643     100000.00000
1970     640000.00000
2433     312500.00000
1407     268000.00000
2042     471000.00000
3329     249000.00000
4589     182805.00000
3624     265000.00000
3754     660000.00000
4157     380000.00000
1124     700000.00000
2692     675000.00000
338     1039000.00000
```

In [155]:

```
# Model Training
lm = LinearRegression()
```

In [156]:

```
#Fitting data in the model
lm.fit(X_train,y_train)
```

Out[156]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fal
se)
```

In [157]:

```
print(lm.intercept_)
```

```
5335032.63650605
```

In [158]:

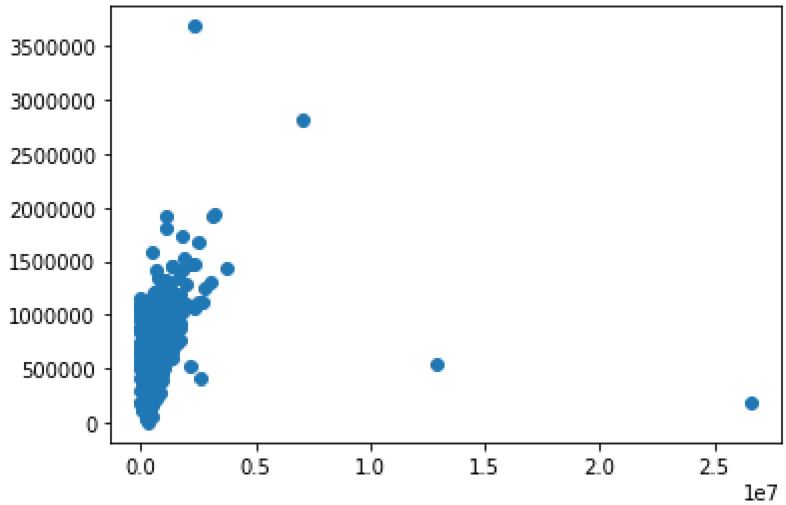
```
predictions = lm.predict(X_test)
```

In [159]:

```
# Scatter plot
plt.scatter(y_test,predictions)
```

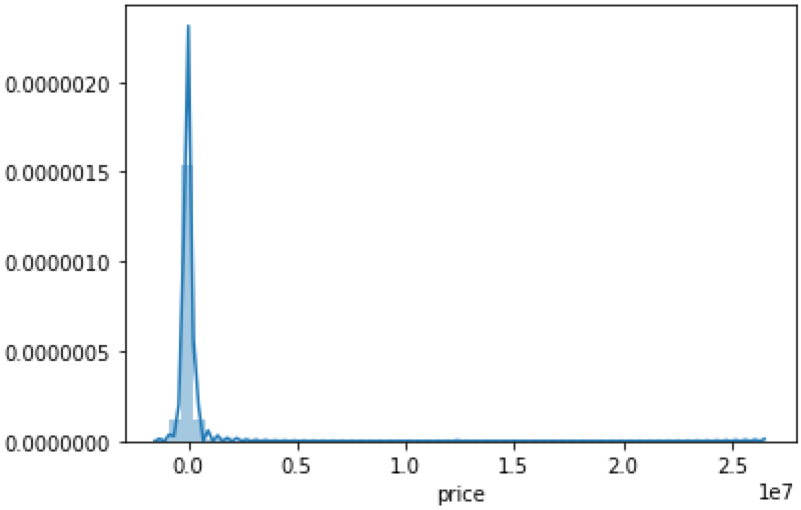
Out[159]:

```
<matplotlib.collections.PathCollection at 0x2537d6f7c88>
```



In [160]:

```
# Plotting Density Distribution
sns.distplot((y_test-predictions),bins=50);
```



In [161]:

```
# Calculating Errors
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 189824.2263705201
MSE: 535920016816.23615
RMSE: 732065.5823191226
```

In [162]:

```
# Scoring provided by linear regression model
model_score =cross_val_score(estimator=LinearRegression(),X=X_train, y=y_train, cv=6)
model_score
```

Out[162]:

```
array([0.51925241, 0.50533965, 0.48512689, 0.55004089, 0.43191235,
       0.48796952])
```

In [167]:

```
# Using GBM algorithm(Gradient Boosting Machine)
model = GradientBoostingRegressor(n_estimators = 400, max_depth = 5, min_samples_split = 2,
          learning_rate = 0.1, loss = 'ls')
model.fit(X_train,y_train)

y_pred = model.predict(X_test)
model.score(X_test,y_test)
```

Out[167]:

```
0.04100223827252125
```

In [184]:

```
# Function to calculate root mean square error
def rmse (y_true,y_pred):
    return np.sqrt(np.mean((y_true - y_pred)**2))
```

In [185]:

```
# Trains model and evaluate model on test data
def model_test(model):
    model.fit(X,y)
    predictions = model.predict(X_test)
    model_rsme = rmse(y_test,predictions)
    return model_rsme
```

In [187]:

```
# K-Nearest Number
knn = KNeighborsRegressor(n_neighbors=5)
knn_rsme = model_test(knn)
print(knn_rsme)
```

```
609133.8329107198
```

In [190]:

```
# Support Vector Machine
svm = SVR()
svm_rsme = model_test(svm)
print(svm_rsme)
```

```
C:\Users\ISHITA GUPTA\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: F
utureWarning: The default value of gamma will change from 'auto' to 'scale'
in version 0.22 to account better for unscaled features. Set gamma explicitl
y to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)

774286.1715799351
```

In [191]:

```
# Descision Tree
tree = DecisionTreeRegressor()
tree_rsme = model_test(tree)
print(tree_rsme)
```

```
21795.201863224007
```