A Project Report Submitted

for the course MA862

in **Statistical Inference Project 2**

by

Ananta Dey(222123009)

Arnab Saha(222123013)

Jaydev Kundu(222123029)

Manish Ray(222123031)

to the

Mathematics Department

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

April 24

Project 2

The objective of this project is to perform an extensive regression analysis using a dataset obtained from an online source. The dataset must contain at least one response variable and a minimum of 10 predictor variables (regressors). This project involve various crucial stages such as data collection, registration, regression analysis, and interpretation of findings.

0.1 Data Collection

Collecting data for regression involves gathering information on both the dependent variable, which you aim to predict or explain, and independent variables, which you believe influence the outcome.

We have choose data for our analysis , named as **'House Price dataset of India'** from kaggle. This data contains 23 regressors (Predicting Parameter) and total 14620 number of data . Regressors are shown in the following table -

Table 1: Description of Chosen Regressors

Regressors	**Description**
ID	Unique identifier for each observation
Date	Date of observation
Number of Bedrooms	Number of bedrooms in the house
Number of Bathrooms	Number of bathrooms in the house
Living Area	Total living area of the house
Lot Area	Total lot area of the property
Number of Floors	Number of floors in the house
Waterfront Present	Indicator variable for waterfront presence
Number of Views	Number of views the property has received
Condition of the House	Condition rating of the house
Grade of the House	Grade rating of the house
Area of the House (excluding basement)	Total area of the house excluding the basement
Area of the Basement	Total area of the basement
Built Year	Year the house was built
Renovation Year	Year of last renovation
Postal Code	Postal code of the property location
Latitude	Latitude coordinate of the property
Longitude	Longitude coordinate of the property
Living Area (Renovated)	Total living area after renovation
Lot Area (Renovated)	Total lot area after renovation
Number of Schools Nearby	Number of schools nearby
Distance from the Airport	Distance of the property from the nearest airport
Price	Sale price of the property

The data we have taken is the following -

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Built Year	Renovation Year	Postal Code	Lattitude	Longitude	living_area_renov	lot_area_renov	Number of schools nearby	Distance from the airport	Price
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4	5	...	1921	0	122003	52.8645	-114.557	2880	5400	2	58	2380000
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0	5	...	1909	0	122004	52.8878	-114.470	2470	4000	2	51	1400000
2	6762810998	42491	5	2.75	2910	9480	1.5	0	0	3	...	1939	0	122004	52.8852	-114.468	2940	6600	1	53	1200000
3	6762812605	42491	4	2.50	3310	42998	2.0	0	0	3	...	2001	0	122005	52.9532	-114.321	3350	42847	3	76	838000
4	6762812919	42491	3	2.00	2710	4500	1.5	0	0	4	...	1929	0	122006	52.9047	-114.485	2060	4500	1	51	805000

5 rows × 23 columns

Figure 1: Dataset

0.2 Analysing data

Before moving to analysis we will start by removing the unnecessary data for less computation. Data with modified regressor are shown as -

	number of bedrooms	number of bathrooms	living area	lot area	number of floors	number of views	condition of the house	grade of the house	Area of the house(excluding basement)	Area of the basement	Built Year	living_area_renov	lot_area_renov	Number of schools nearby	Distance from the airport	Price
number of bedrooms	1.000000	0.509784	0.570526	0.034416	0.177294	0.078665	0.026597	0.352945	0.473599	0.300332	0.152954	0.389855	0.029400	0.003397	-0.006157	0.308460
number of bathrooms	0.509784	1.000000	0.753517	0.080806	0.502924	0.183789	-0.128232	0.663054	0.684391	0.287190	0.498127	0.570530	0.078627	0.002180	0.009206	0.531735
living area	0.570526	0.753517	1.000000	0.174420	0.354743	0.287728	-0.063358	0.761835	0.875793	0.441491	0.309602	0.757571	0.180312	0.002370	0.002511	0.712169
lot area	0.034416	0.080806	0.174420	1.000000	-0.004138	0.078308	-0.008548	0.110546	0.183553	0.019755	0.051615	0.149744	0.706812	-0.012671	0.003291	0.081992
number of floors	0.177294	0.502924	0.354743	-0.004138	1.000000	0.020153	-0.269928	0.463082	0.525643	-0.242976	0.481565	0.285093	-0.010120	-0.007579	0.016567	0.262732
number of views	0.078665	0.183789	0.287728	0.078308	0.020153	1.000000	0.052533	0.254532	0.162672	0.293062	-0.055357	0.281452	0.072300	0.008004	-0.001657	0.395973
condition of the house	0.026597	-0.128232	-0.063358	-0.008548	-0.269928	0.052533	1.000000	-0.152530	-0.167695	0.180609	-0.381718	-0.099743	-0.004748	-0.006939	-0.002136	0.041376
grade of the house	0.352945	0.663054	0.761835	0.110546	0.463082	0.254532	-0.152530	1.000000	0.758222	0.167160	0.440358	0.720019	0.116725	0.000986	0.004940	0.671814
Area of the house(excluding basement)	0.473599	0.684391	0.875793	0.183553	0.525643	0.162672	-0.167695	0.758222	1.000000	-0.046445	0.419369	0.737744	0.194670	-0.002894	0.001222	0.615220
Area of the basement	0.300332	0.287190	0.441491	0.019755	-0.242976	0.293062	0.180609	0.167160	-0.046445	1.000000	-0.138843	0.196403	0.011283	0.010284	0.002926	0.330202
Built Year	0.152954	0.498127	0.309602	0.051615	0.481565	-0.055357	-0.381718	0.440358	0.419369	-0.138843	1.000000	0.328625	0.072874	-0.001631	-0.003968	0.050307
living_area_renov	0.389855	0.570530	0.757571	0.149744	0.285093	0.281452	-0.099743	0.720019	0.737744	0.196403	0.328625	1.000000	0.189225	-0.001203	-0.005673	0.584924
lot_area_renov	0.029400	0.078627	0.180312	0.706812	-0.010120	0.072300	-0.004748	0.116725	0.194670	0.011283	0.072874	0.189225	1.000000	-0.025014	-0.014587	0.075535
Number of schools nearby	0.003397	0.002180	0.002370	-0.012671	-0.007579	0.008004	-0.006939	0.000986	-0.002894	0.010284	-0.001631	-0.001203	-0.025014	1.000000	0.004035	0.009890
Distance from the airport	-0.006157	0.009206	0.002511	0.003291	0.016567	-0.001657	-0.002136	0.004940	0.001222	0.002926	-0.003968	-0.005673	-0.014587	0.004035	1.000000	0.003804
Price	0.308460	0.531735	0.712169	0.081992	0.262732	0.395973	0.041376	0.671814	0.615220	0.330202	0.050307	0.584924	0.075535	0.009890	0.003804	1.000000

Figure 2: These $16th$ data are only of our interest.

We are setting up for data analysis with the above data with the following step and methods

pandas for manipulation, statsmodels for modeling, seaborn for visualization, numpy for computation, StandardScaler for preprocessing.

It initializes two StandardScaler objects, `scaler_x` for features and `scaler_y` for the target variable. Then, it fits the scalers to the data. For `scaler_x`, it fits to the features x, and for `scaler_y`, it fits to the target variable y after reshaping it to a 2D array. After fitting, it transforms the original data using the learned scaling parameters. Finally, it converts the scaled data into pandas DataFrames, `x_scaled` for features and `y_scaled` for the target variable (optional).

0.2.1 Scaling The Data

First we have scaled the data for less computation cost. Both the regressors and dependent variable columns are scaled using Standard scaler.

```
from sklearn.preprocessing import StandardScaler

# Create StandardScaler objects
scaler_x = StandardScaler()
scaler_y = StandardScaler()

# Fit scalers to the data
scaler_x.fit(x)
scaler_y.fit(y.values.reshape(-1, 1))  # Reshape y to a 2D array

# Transform the data
x_scaled = scaler_x.transform(x)
y_scaled = scaler_y.transform(y.values.reshape(-1, 1))

# Convert scaled data to DataFrames (optional)
```

```
x_scaled = pd.DataFrame(x_scaled, columns=x.columns)
y_scaled = pd.DataFrame(y_scaled, columns=["y_scaled"])
```

0.2.2 Fitting The Data

Followed by Linear Regression to fit the model take output as the model coefficients β and the intercept β_0 .

```
from sklearn.linear_model import LinearRegression

# Linear regression object
model = LinearRegression()

# Model fitting
model.fit(x_scaled, y_scaled)

print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

The above model fitting give us the value of the regression coefficient (β_i) as well as the intercept parameter β_0. Values are following -

$$\beta_1 = -0.10894743 \qquad \beta_2 = 0.09719523$$

$$\beta_3 = 0.22889652 \qquad \beta_4 = -0.01618886$$

$$\beta_5 = 0.03631496 \qquad \beta_6 = 0.1446638$$

$$\beta_7 = 0.03939932 \qquad \beta_8 = 0.3731599$$

$$\beta_9 = 0.20203719 \qquad \beta_{10} = 0.0981347$$

$$\beta_{11} = -0.28433728 \qquad \beta_{12} = 0.01686975$$

$$\beta_{13} = -0.03497672 \qquad \beta_{14} = 0.00659034$$

$$\beta_{15} = -0.00250802$$

Intercept(β_0)= -2.80470271e-16

Now with these known coefficient values we will predict the target value , using the code-

```
y_pred = model.predict(x_scaled)
```

Hence we obtained our predicted values as the form of array as - [2.70694928], [1.12738391], [0.63469742], ..., [-1.01712236], [-1.15576853], [-1.30241654]]

0.2.3 Computing SS_{res} and SS_{reg}

We have both the values of actual values (y_i) and the predicted values ($\hat{y}_i$) and we can have mean of actual values ($\bar{y}_i$). Therefore using the following formulas -

$$SS_{res} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$
$$SS_{reg} = \sum_{i=1}^{n}(\hat{y}_i - \bar{y}_i)^2$$

```
y = y_scaled
n = y_pred.shape[0]
ybar = y['y_scaled'].mean()
SSres = 0
SSreg = 0
for i in range(n):
    SSres+=(y['y_scaled'][i]-y_pred[i])**2
    SSreg+=(y_pred[i]-ybar)**2
```

We get the values as -

$$SS_{res} = 5142.469217061956$$
$$SS_{reg} = 9481.460450271394$$

0.2.4 OLS Method for CI

Now to use inbuilt OLS Model to get the confidence interval of the regressor.

```
# Add constant for intercept(beta_0)
```

```
X = sm.add_constant(x_scaled)

# Fitting OLS model
model = sm.OLS(y_scaled, X).fit()

# Print model summary
print(model.summary())
```

With the above method we get the values of Adjusted R-squared, confidence intervals of regressors as shown in the following figure-

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               y_scaled   R-squared:                       0.649
Model:                            OLS   Adj. R-squared:                  0.648
Method:                 Least Squares   F-statistic:                     1925.
Date:                Sat, 27 Apr 2024   Prob (F-statistic):               0.00
Time:                        15:50:12   Log-Likelihood:                -13100.
No. Observations:               14620   AIC:                         2.623e+04
Df Residuals:                   14605   BIC:                         2.634e+04
Df Model:                          14
Covariance Type:            nonrobust
==========================================================================================================
                                            coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------------------
const                                  -1.08e-16      0.005   -2.2e-14      1.000      -0.010       0.010
number of bedrooms                       -0.1089      0.006    -17.485      0.000      -0.121      -0.097
number of bathrooms                       0.0972      0.009     10.951      0.000       0.080       0.115
living area                               0.2289      0.006     40.556      0.000       0.218       0.240
lot area                                 -0.0162      0.007     -2.322      0.020      -0.030      -0.003
number of floors                          0.0363      0.007      5.323      0.000       0.023       0.050
number of views                           0.1447      0.005     26.817      0.000       0.134       0.155
condition of the house                    0.0394      0.005      7.296      0.000       0.029       0.050
grade of the house                        0.3732      0.009     42.078      0.000       0.356       0.391
Area of the house(excluding basement)     0.2020      0.006     33.619      0.000       0.190       0.214
Area of the basement                      0.0981      0.006     17.828      0.000       0.087       0.109
Built Year                               -0.2843      0.007    -43.125      0.000      -0.297      -0.271
living_area_renov                         0.0169      0.008      2.028      0.043       0.001       0.033
lot_area_renov                           -0.0350      0.007     -4.977      0.000      -0.049      -0.021
Number of schools nearby                  0.0066      0.005      1.343      0.179      -0.003       0.016
Distance from the airport                -0.0025      0.005     -0.511      0.609      -0.012       0.007
```

Figure 3: Output of OLS method

Remarks:

(1) Standard Errors assume that the covariance matrix of the errors is correctly specified.

(2) The smallest eigenvalue is 1.17e-27. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

0.2.5 Test for Significance of Regression

Want to test the hypothesis if there is a linear relationship between the response y and any of the regressors $x_1, \ldots, x_n$.

$$H_0 : \beta_1 = \beta_2 = \ldots = \beta_p = 0$$

$$H_1 : \beta_j \neq 0 \text{ for at least one } j$$

The test statistic is

$$F_0 = \frac{\text{SSReg}/p}{\hat{\sigma}^2} = \frac{\text{SSReg}/p}{\text{SSRes}/(n-p-1)} \sim F_{p,n-p-1}, \text{ under } H_0.$$

Reject H_0 iff $F_0 > F_{p,n-p-1;\alpha}$ (at level α)

```
p = x_scaled.shape[1]
F0 = SSreg[0]*(n-p-1)/(SSres[0]*p)
```

We perform this test as mentioned in above code, and we obtained the value F_0=1795.081215801279. The theoretical F0 value is 1.48714, which tell us that to reject null hypothesis.

Remark: Certainly the value is way far then the the cut off points , hence we reject the hypothesis. Means at least one of the regression coefficient(β_i) is non zero. Hence there exist only non linear relation between regressor and the target value.

0.2.6 Hat Matrix calculation for further process

Definition 0.2.1. The fitted value of the response corresponding to regressor values $x = (1, x_1, \ldots, x_p)$ is

$$\hat{y}_b = \beta_{b0} + \beta_{b1}x_1 + \ldots + \beta_{bp}x_p$$

Then $\hat{y} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)$

$$T = X\beta_b = X(X^T X)^{-1} X^T y = Hy,$$

where $H = X(X^T X)^{-1} X^T$ is called the hat-matrix.

We use the following code to evaluate the Hat matirx ($\hat{H}$) as follows-

```
import numpy as np

def hat_matrix(X):
    # Convert X to numpy array
    X = X.values

    # Compute X^TX
    XTX = X.T @ X

    # Compute inverse of XTX
    XTX_inv = np.linalg.inv(XTX)

    # Compute X(X^TX)^{-1}X^T
    H = X @ XTX_inv @ X.T

    return H


H = hat_matrix(x_scaled)
print(H)
```

Figure 4: Hat Matrix code

The output of the Hat matrix ($\hat{H}$) as follows -

```
[[ 3.01971937e-03  4.25378654e-04  3.25132485e-04 ... -3.74893325e-04
   1.85235154e-04 -1.87702082e-04]
 [ 4.04902256e-04  1.12331098e-03  3.53080071e-04 ... -3.00701018e-04
   4.81432766e-05 -2.82584630e-04]
 [ 2.41290564e-04  3.39729001e-04  1.07636781e-03 ... -1.10942149e-04
   8.94707101e-05  7.60514457e-06]
 ...
 [-2.59108220e-04 -2.81647870e-04 -8.58397654e-05 ...  3.44876327e-04
   3.63891667e-05  2.53725393e-04]
 [ 2.63762330e-04  4.40661506e-05  9.40783547e-05 ...  6.52684528e-05
   6.37446544e-04  3.06908299e-04]
 [-1.26552926e-04 -3.11004964e-04  1.48761250e-06 ...  3.09479465e-04
   3.31112092e-04  4.70967890e-04]]
```

Figure 5: Hat matrix output

0.2.7 Residuals, Standardized residual, and Studentized residual

Residual:

$$e_i = y_i - \hat{y}_i \quad \text{for all } i \Rightarrow e = (I - H)y.$$

Standardized residual:

$$d_i = \sqrt{e_i}\frac{1}{\sqrt{MSRes}} \quad \text{for all } i \Rightarrow d = \sqrt{\frac{1}{MSRes}}(I - H)y.$$

Studentized residual:

$$r_i = \frac{\sqrt{e_i}}{\sqrt{MSRes(1 - h_{ii})}} \quad \text{for all } i.$$

We got our Residuals (e_i) as

We got our standardized residuals as

```
residuals
```

```
array([2.30248892, 1.21553137, 1.16402951, ..., 0.11939601, 0.24715842,
       0.23327086])
```

Figure 6: Residuals

```
ss_residuals
```

```
array([3.88226687, 2.04952872, 1.96269053, ..., 0.2013157 , 0.41673813,
       0.39332209])
```

Figure 7: Standardized Residuals

We got our studentized residuals as

```
studentized_residuals
```

```
array([3.88814185, 2.05068082, 1.96374767, ..., 0.20135043, 0.41687102,
       0.39341474])
```

Figure 8: Studentized Residuals

0.2.8 Residual Plots : QQ Plot

The residuals can be assessed for normality using a Q–Q plot. This compares the residuals to "ideal" normal observations. The code for the QQ plots are following -

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
```

```
# Q-Q plot for studentized residuals
qqplot(studentized_residuals, line='s', ax=axes[0])
axes[0].set_title('Q-Q Plot for Studentized Residuals')
# Q-Q plot for residuals
qqplot(residual, line='s', ax=axes[1])
axes[1].set_title('Q-Q Plot for Residuals')

# Q-Q plot for squared residuals
qqplot(np.sqrt(ss_residuals), line='s', ax=axes[2])  # Taking square root of ss_resid
axes[2].set_title('Q-Q Plot for Square Root of SS Residuals')

plt.tight_layout()
plt.show()
```

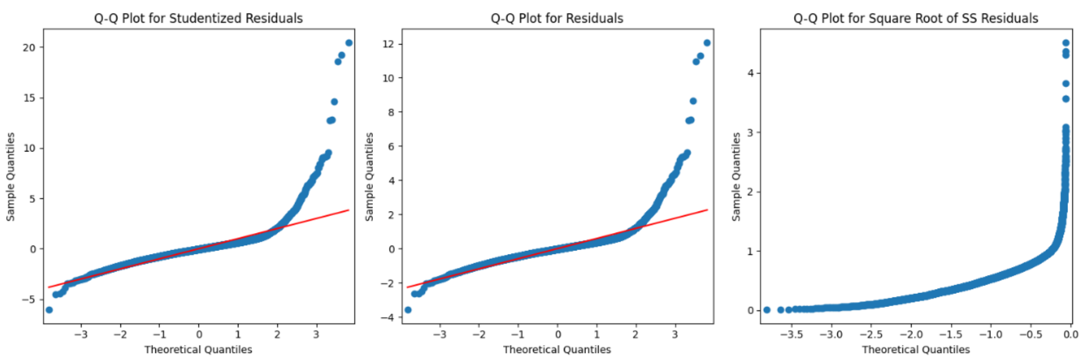

Figure 9: QQ Plot

0.2.9 Plot of residual against fitted values

```
# Plot of residual against fitted values
plt.figure(figsize=(10, 5))
plt.scatter(y_pred, residuals, alpha=0.5)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel("Fitted values")
plt.ylabel("Residuals")
plt.title("Residual Plot: Residuals vs Fitted Values")
plt.show()
```

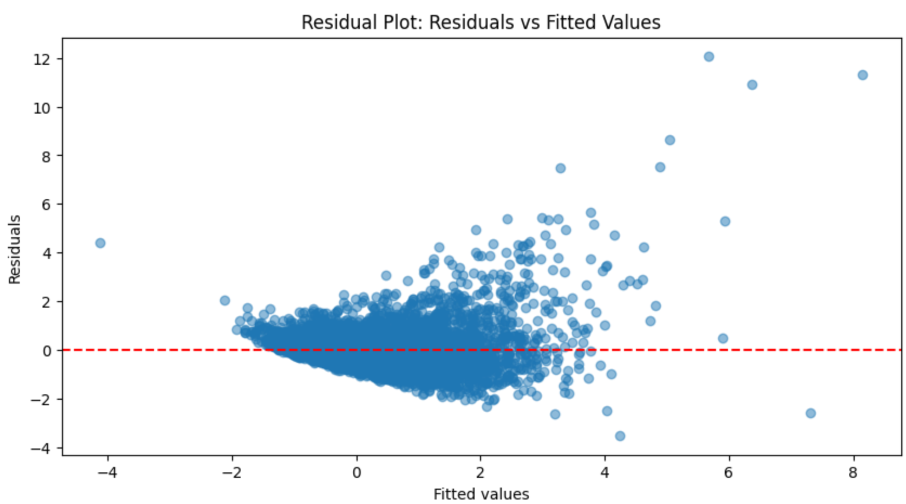

Figure 10: Residuals Plot: Residuals vs Fitted Values

0.2.10 Residual plot vs Regressors

For this purpose we use the following code to get residual plots for different kind of combination of regressors -

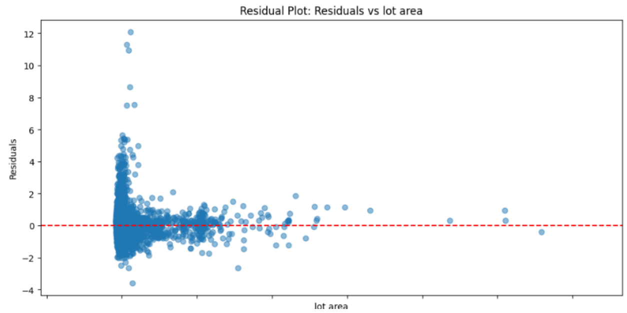

Figure 11: Residuals Plot: Residuals vs lot area

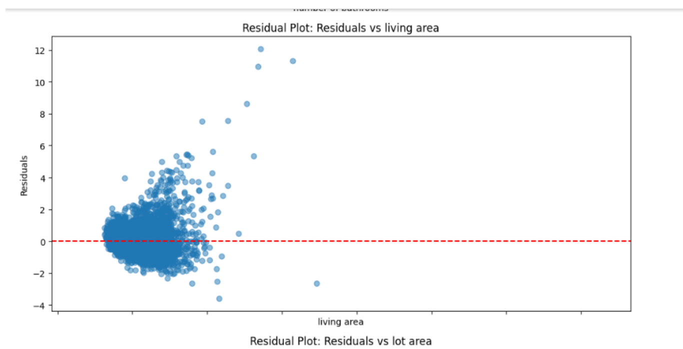

Figure 12: residuals vs living area

0.2.11 Partial Regression Plot

y is regressed on $x_1, \dots, x_k$ except x_j. x_j is regressed on $x_1, \dots, x_k$ except x_j. Plot y residual, $e_i(y|x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_k)$ against x_j residual, $e_i(x_j|x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_k)$. For the ideal scenario, the partial regression plot should show a linear relationship (straight line with non-zero slope).

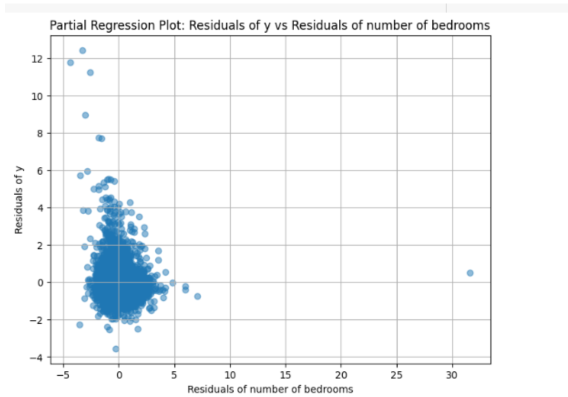

Figure 13: Partial Regression plot : Residuals of y vs residuals of number of bedrooms

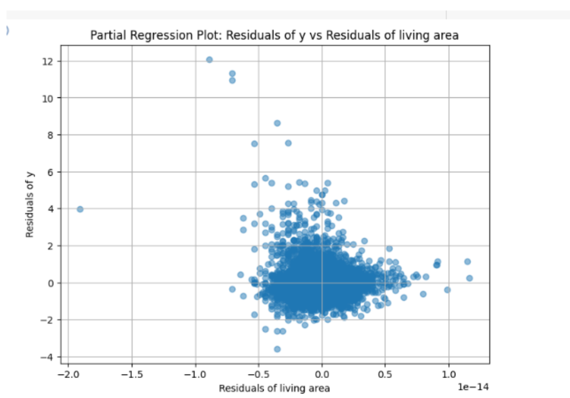

Figure 14: Partial regression plot : residuals of y vs residuals of living area

0.3 Subset Selection

0.3.1 All Possible Selection method

The **all possible regression** function systematically explores all possible combinations of predictor variables to identify the best regression model based on adjusted R-squared score. It initializes variables to store the best model and the highest score found so far. Using nested loops, it iterates through different numbers of predictors and their combinations. For each combination, it creates a subset of predictor variables and fits an Ordinary Least Squares (OLS) regression model to the data. The function updates the best model and score if the adjusted R-squared score of the current model is higher than the previous best score. Finally, it returns the best model found after exploring all possible combinations. This exhaustive search approach ensures that the model with the highest explanatory power is selected among all potential combinations of predictors.

```
import itertools
import statsmodels.api as sm

def all_possible_regression(X, y):
    best_model = None
    best_score = float('-inf')

    for k in range(1, len(X.columns) + 1):
        for subset in itertools.combinations(X.columns, k):
            X_subset = X[list(subset)]
            X_subset = sm.add_constant(X_subset)
```

```
            model = sm.OLS(y, X_subset).fit()
            if model.rsquared_adj > best_score:
                best_model = model
                best_score = model.rsquared_adj

    return best_model
```

0.3.2 Forward Selection

The forward selection function implements a method for iteratively building a regression model by selecting predictors based on their individual performance. It starts by initializing variables such as the set of remaining predictors and the list of selected predictors. Within a loop, the function evaluates each remaining predictor's contribution to the model's performance by fitting models that include the selected predictors along with the current predictor being assessed. The predictor that results in the highest improvement in adjusted R-squared compared to the previous iteration is selected and added to the list of chosen predictors. The loop continues until the improvement in adjusted R-squared falls below a predefined threshold. Once the selection process is complete, the function fits a final regression model using the selected predictors. The function returns both the best-fitted model obtained through forward selection and the list of predictors chosen by the algorithm. This approach allows for the construction of a parsimonious regression model that includes only the most informative predictors, enhancing interpretability and potentially improving predictive performance.

The code of the above is given by,

```
def forward_selection(X, y):
```

```
remaining_predictors = set(X.columns)
selected_predictors = []
best_score = float('-inf')
old_score = 0

while remaining_predictors:
    scores = []
    for predictor in remaining_predictors:
        model = sm.OLS(y, sm.add_constant(X[selected_predictors + [predictor]])).
        scores.append((predictor, model.rsquared_adj))

    best_predictor, best_score = max(scores, key=lambda x: x[1])
    if best_score - old_score >1e-03:
      selected_predictors.append(best_predictor)
      remaining_predictors.remove(best_predictor)
      old_score = best_score
    else:
      break



return sm.OLS(y, sm.add_constant(X[selected_predictors])).fit(),selected_predicto
```

What we observe is that when we model our data using the following regressors:

- Living Area
- Grade of the House

- Year Built
- Number of Views
- Number of Bedrooms
- Number of Bathrooms
- Lot Area Renovated

our model performs better, as evidenced by the adjusted R^2 value of 0.648, which is the highest among all other combinations of regressors.

```
x_forward.head()
```

	living area	grade of the house	Built Year	number of views	number of bedrooms	number of bathrooms	lot_area_renov
0	1.671691	1.972420	-1.692844	4.916126	1.726515	0.481119	-0.282203
1	0.885260	0.270281	-2.099726	-0.304223	0.661197	0.481119	-0.335930
2	0.874487	0.270281	-1.082522	-0.304223	1.726515	0.805833	-0.236151
3	1.305408	1.121351	1.019699	-0.304223	0.661197	0.481119	1.154887
4	0.659026	0.270281	-1.421590	-0.304223	-0.404121	-0.168309	-0.316742

```
[ ] x_forward.corr()
```

	living area	grade of the house	Built Year	number of views	number of bedrooms	number of bathrooms	lot_area_renov
living area	1.000000	0.761835	0.309602	0.287728	0.570526	0.753517	0.180312
grade of the house	0.761835	1.000000	0.440358	0.254532	0.352945	0.663054	0.116725
Built Year	0.309602	0.440358	1.000000	-0.055357	0.152954	0.498127	0.072874
number of views	0.287728	0.254532	-0.055357	1.000000	0.078665	0.183789	0.072300
number of bedrooms	0.570526	0.352945	0.152954	0.078665	1.000000	0.509784	0.029400
number of bathrooms	0.753517	0.663054	0.498127	0.183789	0.509784	1.000000	0.078627
lot_area_renov	0.180312	0.116725	0.072874	0.072300	0.029400	0.078627	1.000000

Figure 15: Results

0.3.3 Results

In the forward selection method, the adjusted R^2 value for the best model ($R^2_{\text{adj}} = 0.646$) is slightly lower compared to the adjusted R^2 value for the model with all possible regressors ($R^2_{\text{adj}} = 0.648$). Specifically, the best model achieved an adjusted R^2 value of 0.646, while the model with all possible regressors attained an adjusted R^2 value of 0.648.

0.3.4 Multicolinearity

For multicolinearity check, we used seaborn hitmap plot and checked the covariance between regressors which are greater than 0.7. And we see that Living area with grade of the house and with number of bathrooms column has significant correlation.

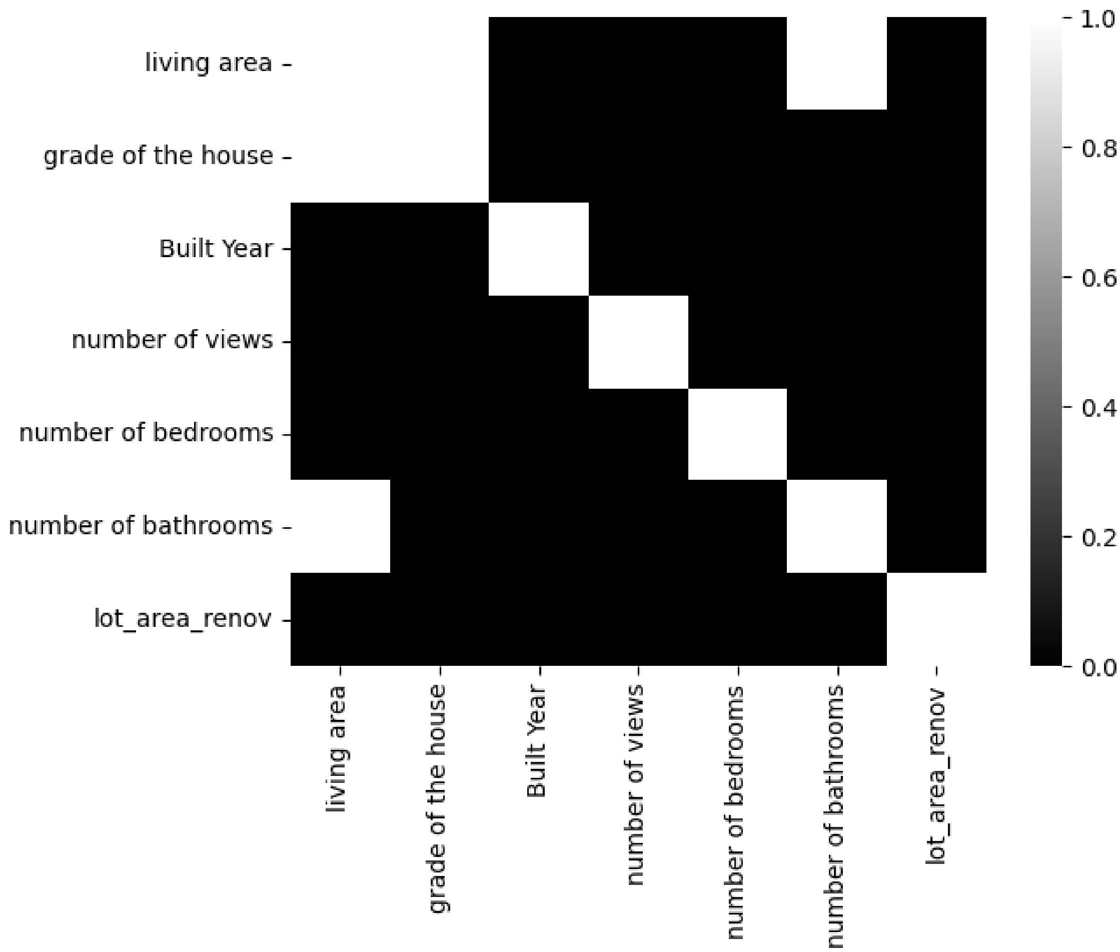

Figure 16: Living area with grade of the house and with number of bathrooms column has significant correlation