```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('dark_background')
import warnings
warnings.filterwarnings('ignore')


#Beginning data cleaning
data = pd.read_csv('/content/drive/MyDrive/Notebooks/Raw_Housing_Prices.csv')
data.head()
```

	ID	Date House was Sold	Sale Price	No of Bedrooms	No of Bathrooms	Flat Area (in Sqft)	Lot Area (in Sqft)	No of Floors	Waterfront View	No of Times Visited	Condition of the House	Over Gr
0	7129300520	14 October 2017	221900.0	3	1.00	1180.0	5650.0	1.0	No	None	Fair	
1	6414100192	14 December 2017	538000.0	3	2.25	2570.0	7242.0	2.0	No	None	Fair	
2	5631500400	15 February 2016	180000.0	2	1.00	770.0	10000.0	1.0	No	None	Fair	
3	2487200875	14 December 2017	604000.0	4	3.00	1960.0	5000.0	1.0	No	None	Excellent	
4	1954400510	15 February 2016	510000.0	3	2.00	1680.0	8080.0	1.0	No	None	Fair	

```
data['Sale Price'].describe()
```

```
count    2.160900e+04
mean     5.116186e+05
std      2.500620e+05
min      7.500000e+04
25%      3.219500e+05
50%      4.500000e+05
75%      6.450000e+05
max      1.129575e+06
Name: Sale Price, dtype: float64
```

```
data['Sale Price'].plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb7fc29cd68>
```

```
# checking quantiles
q1 = data['Sale Price'].quantile(0.25)
q3 = data['Sale Price'].quantile(0.75)
q1, q3
```

```
(321950.0, 645000.0)
```

```
#calculating iqr
```

```
iqr = q3 - q1
iqr
```

```
323050.0
```

```
upper_limit = q3 + 1.5*iqr
lower_limit = q1 - 1.5*iqr
upper_limit, lower_limit
```

```
(1129575.0, -162625.0)
```

```
# treating outliers
def limit_imputer(value):
  if value > upper_limit:
    return upper_limit
  if value < lower_limit:
    return lower_limit
  else:
    return value
```

```
data['Sale Price'] = data['Sale Price'].apply(limit_imputer)
```

```
data['Sale Price'].plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb7fc181748>
```

```
#checking missing values
data.isnull().sum()
```

```
ID                                          0
Date House was Sold                         0
Sale Price                                  0
No of Bedrooms                              0
No of Bathrooms                             0
Flat Area (in Sqft)                         0
Lot Area (in Sqft)                          0
No of Floors                                0
Waterfront View                             0
No of Times Visited                         0
Condition of the House                      0
Overall Grade                               0
Area of the House from Basement (in Sqft)   0
Basement Area (in Sqft)                     0
Age of House (in Years)                     0
Renovated Year                              0
Zipcode                                     0
Latitude                                    0
Longitude                                   0
Living Area after Renovation (in Sqft)      0
Lot Area after Renovation (in Sqft)         0
dtype: int64
```

```
data['Sale Price'].dropna(inplace=True)
data["Sale Price"].isnull().sum()
```

```
0
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21609 entries, 0 to 21608
Data columns (total 21 columns):
```

```
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   ID                                        21609 non-null  int64
 1   Date House was Sold                       21609 non-null  object
 2   Sale Price                                21609 non-null  float64
 3   No of Bedrooms                            21609 non-null  int64
 4   No of Bathrooms                           21609 non-null  float64
 5   Flat Area (in Sqft)                       21609 non-null  float64
 6   Lot Area (in Sqft)                        21609 non-null  float64
 7   No of Floors                              21609 non-null  float64
 8   Waterfront View                           21609 non-null  object
 9   No of Times Visited                       21609 non-null  object
 10  Condition of the House                    21609 non-null  object
 11  Overall Grade                             21609 non-null  int64
 12  Area of the House from Basement (in Sqft)  21609 non-null  float64
 13  Basement Area (in Sqft)                   21609 non-null  int64
 14  Age of House (in Years)                   21609 non-null  int64
 15  Renovated Year                            21609 non-null  int64
 16  Zipcode                                   21609 non-null  float64
 17  Latitude                                  21609 non-null  float64
 18  Longitude                                 21609 non-null  float64
 19  Living Area after Renovation (in Sqft)    21609 non-null  float64
 20  Lot Area after Renovation (in Sqft)       21609 non-null  int64
dtypes: float64(10), int64(7), object(4)
memory usage: 3.5+ MB
```

```
#isolating numerical variables
numerical_columns = ['No of Bathrooms', 'Flat Area (in Sqft)','Lot Area (in Sqft)',
                     'Area of the House from Basement (in Sqft)','Latitude',
                     'Longitude','Living Area after Renovation (in Sqft)']
```

```
#imputing missing values
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values = np.nan, strategy = 'median')
data[numerical_columns] = imputer.fit_transform(data[numerical_columns])
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 21609 entries, 0 to 21608
Data columns (total 21 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   ID                                      21609 non-null  int64  
 1   Date House was Sold                     21609 non-null  object 
 2   Sale Price                              21609 non-null  float64
 3   No of Bedrooms                          21609 non-null  int64  
 4   No of Bathrooms                         21609 non-null  float64
 5   Flat Area (in Sqft)                     21609 non-null  float64
 6   Lot Area (in Sqft)                      21609 non-null  float64
 7   No of Floors                            21609 non-null  float64
 8   Waterfront View                         21609 non-null  object 
 9   No of Times Visited                     21609 non-null  object 
 10  Condition of the House                  21609 non-null  object 
 11  Overall Grade                           21609 non-null  int64  
 12  Area of the House from Basement (in Sqft)  21609 non-null  float64
 13  Basement Area (in Sqft)                 21609 non-null  int64  
 14  Age of House (in Years)                 21609 non-null  int64  
 15  Renovated Year                          21609 non-null  int64  
 16  Zipcode                                 21609 non-null  float64
 17  Latitude                                21609 non-null  float64
 18  Longitude                               21609 non-null  float64
 19  Living Area after Renovation (in Sqft)  21609 non-null  float64
 20  Lot Area after Renovation (in Sqft)     21609 non-null  int64  
dtypes: float64(10), int64(7), object(4)
memory usage: 3.5+ MB
```

```
#treating zipcaodes
imputer = SimpleImputer(missing_values = np.nan, strategy = 'most_frequent')
data['Zipcode'] = imputer.fit_transform(data['Zipcode'].values.reshape(-1,1))
```

```
data['Zipcode'].shape
```

```
(21609,)
```

```
column = data["Zipcode"].values.reshape(-1,1)
column.shape
```

```
(21609, 1)
```

```
imputer = SimpleImputer(missing_values = np.nan, strategy = 'most_frequent')
data['Zipcode'] = imputer.fit_transform(column)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21609 entries, 0 to 21608
Data columns (total 21 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   ID                                      21609 non-null  int64  
 1   Date House was Sold                     21609 non-null  object 
 2   Sale Price                              21609 non-null  float64
 3   No of Bedrooms                          21609 non-null  int64  
 4   No of Bathrooms                         21609 non-null  float64
 5   Flat Area (in Sqft)                     21609 non-null  float64
 6   Lot Area (in Sqft)                      21609 non-null  float64
 7   No of Floors                            21609 non-null  float64
 8   Waterfront View                         21609 non-null  object 
 9   No of Times Visited                     21609 non-null  object 
 10  Condition of the House                  21609 non-null  object 
 11  Overall Grade                           21609 non-null  int64  
 12  Area of the House from Basement (in Sqft)  21609 non-null  float64
 13  Basement Area (in Sqft)                 21609 non-null  int64  
 14  Age of House (in Years)                 21609 non-null  int64  
 15  Renovated Year                          21609 non-null  int64  
 16  Zipcode                                 21609 non-null  float64
 17  Latitude                                21609 non-null  float64
 18  Longitude                               21609 non-null  float64
 19  Living Area after Renovation (in Sqft)  21609 non-null  float64
 20  Lot Area after Renovation (in Sqft)     21609 non-null  int64  
dtypes: float64(10), int64(7), object(4)
memory usage: 3.5+ MB
```

```
#treating times visited
data['No of Times Visited'].unique()
```

```
array(['None', 'Thrice', 'Four', 'Twice', 'Once'], dtype=object)
```

```
# converting from string to categorical
mapping = {'None' : "0",
           'Once' : '1',
           'Twice' : '2',
           'Thrice' : '3',
           'Four' : '4'}

data['No of Times Visited'] = data['No of Times Visited'].map(mapping)


data['No of Times Visited'].unique()
```

```
array(['0', '3', '4', '2', '1'], dtype=object)
```

```
# new variable creation
data['Ever Renovated'] = np.where(data['Renovated Year'] == 0, 'No', 'Yes')


data.head()
```

ID	Date House was Sold	Sale Price	No of Bedrooms	No of Bathrooms	Flat Area (in Sqft)	Lot Area (in Sqft)	No of Floors	Waterfront View	No of Times Visited	Condition of the House	Over Gr

```
#manipulating datetime variable
data['Purchase Year'] = pd.DatetimeIndex(data['Date House was Sold']).year
```

```
data['Years Since Renovation'] = np.where(data['Ever Renovated'] == 'Yes',
                                          abs(data['Purchase Year'] -
                                              data['Renovated Year']), 0)
```

```
data.head()
```

	ID	Date House was Sold	Sale Price	No of Bedrooms	No of Bathrooms	Flat Area (in	Lot Area (in	No of Floors	Waterfront View	No of Times Visited	Condition of the House	Over Gr

```
# dropping redundant variables
data.drop( columns = ['Purchase Year', 'Date House was Sold', 'Renovated Year'], inplace = True)
```

```
                        14
```

```
data.head()
```

	ID	Sale Price	No of Bedrooms	No of Bathrooms	Flat Area (in Sqft)	Lot Area (in Sqft)	No of Floors	Waterfront View	No of Times Visited	Condition of the House	Overall Grade	Area Ho f Basem Sq
0	7129300520	221900.0	3	1.00	1180.0	5650.0	1.0	No	0	Fair	7	118
1	6414100192	538000.0	3	2.25	2570.0	7242.0	2.0	No	0	Fair	7	217
2	5631500400	180000.0	2	1.00	770.0	10000.0	1.0	No	0	Fair	6	77
3	2487200875	604000.0	4	3.00	1960.0	5000.0	1.0	No	0	Excellent	7	105
4	1954400510	510000.0	3	2.00	1680.0	8080.0	1.0	No	0	Fair	8	168

```
data.drop( columns = 'ID', inplace = True)
```

```
data['Condition of the House'].head(10)
```

```
0         Fair
1         Fair
2         Fair
3    Excellent
4         Fair
5         Fair
```

```
6         Fair
7         Fair
8         Fair
9         Fair
Name: Condition of the House, dtype: object
```

```
data['Condition of the House'].value_counts()
```

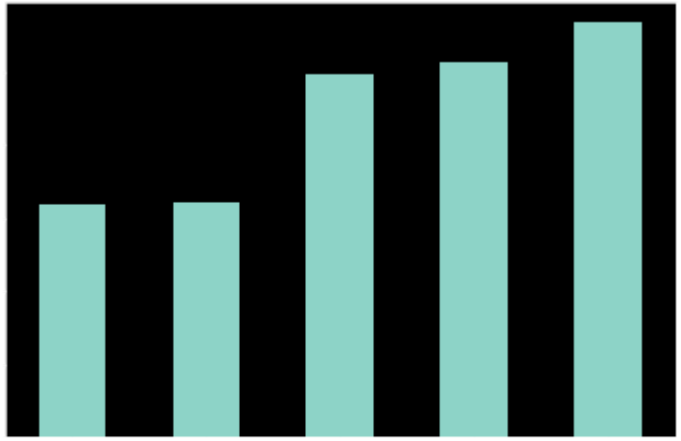
```
Fair         14028
Good          5678
Excellent     1701
Okay           172
Bad             30
Name: Condition of the House, dtype: int64
```

```
data.groupby('Condition of the House')['Sale Price'].mean().plot(kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb7ee4ba5c0>
```

```
data.groupby('Condition of the House')['Sale Price'].mean().sort_values().plot(kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb7ece7abe0>
```



```
data.groupby('Waterfront View')['Sale Price'].mean().sort_values().plot(kind = 'bar')
```

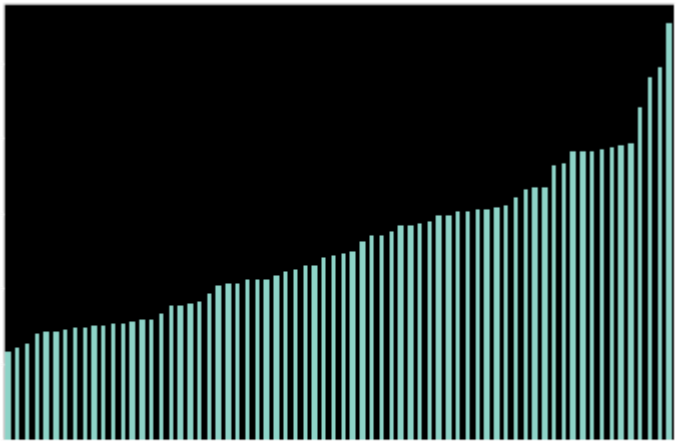
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb7fbcdff28>
```

```
data.groupby('Ever Renovated')['Sale Price'].mean().sort_values().plot(kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb7ecdbcfd0>
```

```
data.groupby('Zipcode',)['Sale Price'].mean().sort_values().plot(kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb7ece7a128>
```

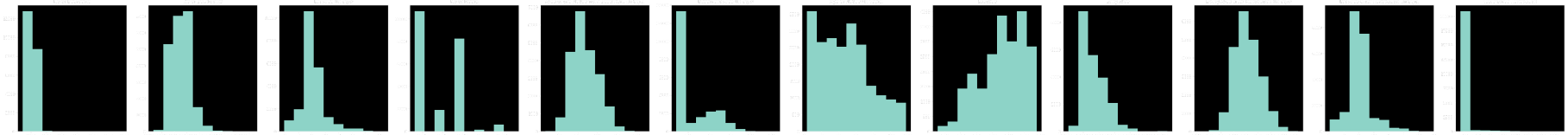


```
#Beginning Linear Regression
```

```
data.dropna(inplace=True)
X = data.drop(columns=['Sale Price'])
Y = data['Sale Price']


#checking distribution of independent numerical variables
def distribution(data ,var):
  plt.figure(figsize = (len(var)*6,6), dpi = 120)
  for j,i in enumerate(var):
    plt.subplot(1,len(var),j+1)
    plt.hist(data[i])
    plt.title(i)


numerical_columns = ['No of Bedrooms', 'No of Bathrooms', 'Lot Area (in Sqft)',
       'No of Floors',
       'Area of the House from Basement (in Sqft)', 'Basement Area (in Sqft)',
       'Age of House (in Years)', 'Latitude', 'Longitude',
       'Living Area after Renovation (in Sqft)',
       'Lot Area after Renovation (in Sqft)',
       'Years Since Renovation']


for i in numerical_columns:
  X[i] = pd.to_numeric(X[i])


distribution(X, numerical_columns)
```

```
#removing right skew
def right_skew(x):
  return np.log(abs(x+500))
```

```
right_skew_variables = ['No of Bedrooms', 'No of Bathrooms', 'Lot Area (in Sqft)',
       'No of Floors',
       'Area of the House from Basement (in Sqft)', 'Basement Area (in Sqft)',
        'Longitude',
       'Living Area after Renovation (in Sqft)',
       'Lot Area after Renovation (in Sqft)',
       'Years Since Renovation']



for i in right_skew_variables:
  X[i] = X[i].map(right_skew)

# removing infinite values
X = X.replace(np.inf, np.nan)
X.dropna(inplace=True)


distribution(X, numerical_columns)
```



```
X.head()
```

	No of Bedrooms	No of Bathrooms	Flat Area (in Sqft)	Lot Area (in Sqft)	No of Floors	Waterfront View	No of Times Visited	Condition of the House	Overall Grade	Area of the House from Basement (in	Basement Area (in Sqft)	Age Hou (Year

```
X["Waterfront View"] = X["Waterfront View"].map({    'No':0,
   'Yes':1
})


X['Condition of the House'] = X['Condition of the House'].map({'Bad':1,
                                                               'Okay':2,
                                                               'Fair':3,
                                                               'Good':4,
                                                               'Excellent':5
})

X['Ever Renovated'] = X['Ever Renovated'].map({
    'No':0,
    'Yes':1
})

X.head()
```

	No of	No of	Flat Area	Lot Area	No of	Waterfront	No of	Condition	Overall	Area of the House	Basement	Age
												Hou

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
Y = data['Sale Price']
X1 = scaler.fit_transform(X)
X = pd.DataFrame(data = X1, columns = X.columns)
X.head()
```

	No of Bedrooms	No of Bathrooms	Flat Area (in Sqft)	Lot Area (in Sqft)	No of Floors	Waterfront View	No of Times Visited	Condition of the House	Overall Grade	Area of the House from Basement (in Sqft)	Basement Area (in Sqft)
0	-0.398647	-1.448931	-0.979905	-0.412101	-0.915604	-0.087181	-0.30579	-0.629203	-0.563993	-0.767588	-0.726332
1	-0.398647	0.176496	0.533718	-0.139474	0.937193	-0.087181	-0.30579	-0.629203	-0.563993	0.642025	0.538457
2	-1.477788	-1.448931	-1.426369	0.221390	-0.915604	-0.087181	-0.30579	-0.629203	-1.468566	-1.618851	-0.726332
3	0.678352	1.149811	-0.130534	-0.544388	-0.915604	-0.087181	-0.30579	2.444136	-0.563993	-1.012639	1.504500
4	-0.398647	-0.148266	-0.435436	-0.017762	-0.915604	-0.087181	-0.30579	-0.629203	0.340581	0.025117	-0.726332

```
X.corr()
```

No of Bathrooms	0.516645	1.000000	0.754415	0.105010	0.500980	0.063683	0.187657	-0.124874	0.635778	0.696041
Flat Area (in Sqft)	0.577469	0.754415	1.000000	0.341571	0.354268	0.103841	0.284678	-0.058922	0.705725	0.853690
Lot Area (in Sqft)	0.175425	0.105010	0.341571	1.000000	-0.218404	0.074316	0.121698	0.066113	0.165808	0.319715
No of Floors	0.175995	0.500980	0.354268	-0.218404	1.000000	0.023721	0.029503	-0.263676	0.461442	0.548408
Waterfront View	-0.006617	0.063683	0.103841	0.074316	0.023721	1.000000	0.401856	0.016650	0.070332	0.063294
No of Times Visited	0.079649	0.187657	0.284678	0.121698	0.029503	0.401856	1.000000	0.045978	0.223661	0.161106
Condition of the House	0.028514	-0.124874	-0.058922	0.066113	-0.263676	0.016650	0.045978	1.000000	-0.143747	-0.153588
Overall Grade	0.349933	0.635778	0.705725	0.165808	0.461442	0.070332	0.223661	-0.143747	1.000000	0.723789
Area of the House from Basement (in Sqft)	0.509475	0.696041	0.853690	0.319715	0.548408	0.063294	0.161106	-0.153588	0.723789	1.000000
Basement Area (in Sqft)	0.276781	0.254042	0.373296	0.056278	-0.266598	0.063276	0.249446	0.176043	0.116078	-0.111289
Age of House (in Years)	-0.154613	-0.506206	-0.318146	-0.006100	-0.489232	0.026149	0.053395	0.361383	-0.456711	-0.448692
Zipcode	-0.153163	-0.204097	-0.199380	-0.279267	-0.059289	0.030286	0.084830	0.003076	-0.185844	-0.285278
Latitude	-0.008867	0.024506	0.052538	-0.145945	0.049640	-0.014275	0.006162	-0.015008	0.111226	-0.015269
Longitude	0.129997	0.223332	0.240124	0.376102	0.125724	-0.041934	-0.078472	-0.106546	0.201765	0.360187

Living Area after Renovation (in Sqft)	0.404806	0.572407	0.739515	0.361633	0.277817	0.080573	0.268524	-0.090182	0.676795	0.720753
Lot Area after Renovation (in Sqft)	0.154329	0.095798	0.318573	0.917835	-0.210687	0.083232	0.118788	0.073370	0.167561	0.301163
Ever Renovated	0.018555	0.050239	0.055111	0.022789	0.006318	0.093291	0.104051	-0.060152	0.010010	0.026070
Years Since Renovation	-0.006734	0.004644	0.024552	0.033036	-0.000379	0.105822	0.094621	-0.012115	-0.023473	0.013457

```
## pair of independent variables with correlation greater than 0.5
k = X.corr()
z = [[str(i),str(j)] for i in k.columns for j in k.columns if (k.loc[i,j] >abs(0.5))&(i!=j)]
z, len(z)
```

```
([['No of Bedrooms', 'No of Bathrooms'],
  ['No of Bedrooms', 'Flat Area (in Sqft)'],
  ['No of Bedrooms', 'Area of the House from Basement (in Sqft)'],
  ['No of Bathrooms', 'No of Bedrooms'],
  ['No of Bathrooms', 'Flat Area (in Sqft)'],
  ['No of Bathrooms', 'No of Floors'],
  ['No of Bathrooms', 'Overall Grade'],
  ['No of Bathrooms', 'Area of the House from Basement (in Sqft)'],
  ['No of Bathrooms', 'Living Area after Renovation (in Sqft)'],
  ['Flat Area (in Sqft)', 'No of Bedrooms'],
  ['Flat Area (in Sqft)', 'No of Bathrooms'],
  ['Flat Area (in Sqft)', 'Overall Grade'],
  ['Flat Area (in Sqft)', 'Area of the House from Basement (in Sqft)'],
  ['Flat Area (in Sqft)', 'Living Area after Renovation (in Sqft)'],
  ['Lot Area (in Sqft)', 'Lot Area after Renovation (in Sqft)'],
  ['No of Floors', 'No of Bathrooms'],
  ['No of Floors', 'Area of the House from Basement (in Sqft)'],
  ['Overall Grade', 'No of Bathrooms'],
  ['Overall Grade', 'Flat Area (in Sqft)'],
```

```
          ['Overall Grade', 'Area of the House from Basement (in Sqft)'],
          ['Overall Grade', 'Living Area after Renovation (in Sqft)'],
          ['Area of the House from Basement (in Sqft)', 'No of Bedrooms'],
          ['Area of the House from Basement (in Sqft)', 'No of Bathrooms'],
          ['Area of the House from Basement (in Sqft)', 'Flat Area (in Sqft)'],
          ['Area of the House from Basement (in Sqft)', 'No of Floors'],
          ['Area of the House from Basement (in Sqft)', 'Overall Grade'],
          ['Area of the House from Basement (in Sqft)',
           'Living Area after Renovation (in Sqft)'],
          ['Living Area after Renovation (in Sqft)', 'No of Bathrooms'],
          ['Living Area after Renovation (in Sqft)', 'Flat Area (in Sqft)'],
          ['Living Area after Renovation (in Sqft)', 'Overall Grade'],
          ['Living Area after Renovation (in Sqft)',
           'Area of the House from Basement (in Sqft)'],
          ['Lot Area after Renovation (in Sqft)', 'Lot Area (in Sqft)'],
          ['Ever Renovated', 'Years Since Renovation'],
          ['Years Since Renovation', 'Ever Renovated']],
         34)


# Importing Variance_inflation_Factor funtion from the Statsmodels
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif_data = X[:]

## Calculating VIF for every column
VIF = pd.Series([variance_inflation_factor(vif_data.values, i) for i in range(vif_data.shape[1])], index = vif_data.col
VIF
```

```
No of Bedrooms                              1.736931
No of Bathrooms                             3.424393
Flat Area (in Sqft)                        21.514533
Lot Area (in Sqft)                          6.844926
No of Floors                                2.388708
Waterfront View                             1.211015
No of Times Visited                         1.415596
Condition of the House                      1.260549
Overall Grade                               2.905865
Area of the House from Basement (in Sqft)  23.289239
Basement Area (in Sqft)                     6.561328
Age of House (in Years)                     2.458302
Zipcode                                     1.668833
```

```
Latitude                                  1.191495
Longitude                                 1.880317
Living Area after Renovation (in Sqft)    2.917259
Lot Area after Renovation (in Sqft)       6.603083
Ever Renovated                            3.022760
Years Since Renovation                    2.872050
dtype: float64
```

```
def MC_remover(data):
  vif = pd.Series([variance_inflation_factor(data.values, i) for i in range(data.shape[1])], index = data.columns)
  if vif.max() > 5:
    print(vif[vif == vif.max()].index[0],'has been removed')
    data = data.drop(columns = [vif[vif == vif.max()].index[0]])
    return data
  else:
    print('No Multicollinearity present anymore')
    return data


for i in range(7):
  vif_data = MC_remover(vif_data)

vif_data.head()
```

```
Area of the House from Basement (in Sqft) has been removed
Lot Area (in Sqft) has been removed
Flat Area (in Sqft) has been removed
No Multicollinearity present anymore
No Multicollinearity present anymore
```

```
# Calculating VIF for remaining columns
VIF = pd.Series([variance_inflation_factor(vif_data.values, i) for i in range(vif_data.shape[1])], index = vif_data.col
VIF, len(vif_data.columns)
```

```
(No of Bedrooms                          1.498157
 No of Bathrooms                         2.950107
 No of Floors                            2.185237
 Waterfront View                         1.209171
 No of Times Visited                     1.410593
 Condition of the House                  1.253804
 Overall Grade                           2.541427
 Basement Area (in Sqft)                 1.639834
 Age of House (in Years)                 2.392439
 Zipcode                                 1.666011
 Latitude                                1.183389
 Longitude                               1.857951
 Living Area after Renovation (in Sqft)  2.503466
 Lot Area after Renovation (in Sqft)     1.552630
 Ever Renovated                          3.017581
 Years Since Renovation                  2.868474
 dtype: float64, 16)
```

```
X = vif_data[:]
```

```
Y = data['Sale Price']
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 101)

x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((15126, 16), (6483, 16), (15126,), (6483,))
```

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression(normalize = True)
lr.fit(x_train, y_train)
```

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

```
lr.coef_
```

```
array([  1584.86669312,  42601.67421626,  23310.57525319,   9989.31321342,
        30480.01914579,  16059.91826464, 108934.75599668,  11330.36808308,
        65047.79835963, -15608.43858763,  75617.37499127,  -7749.57798632,
        54292.09103532,   2016.74778297,  16444.49611058, -11320.49390293])
```

```
predictions = lr.predict(x_test)
```

```
lr.score(x_test, y_test)
```

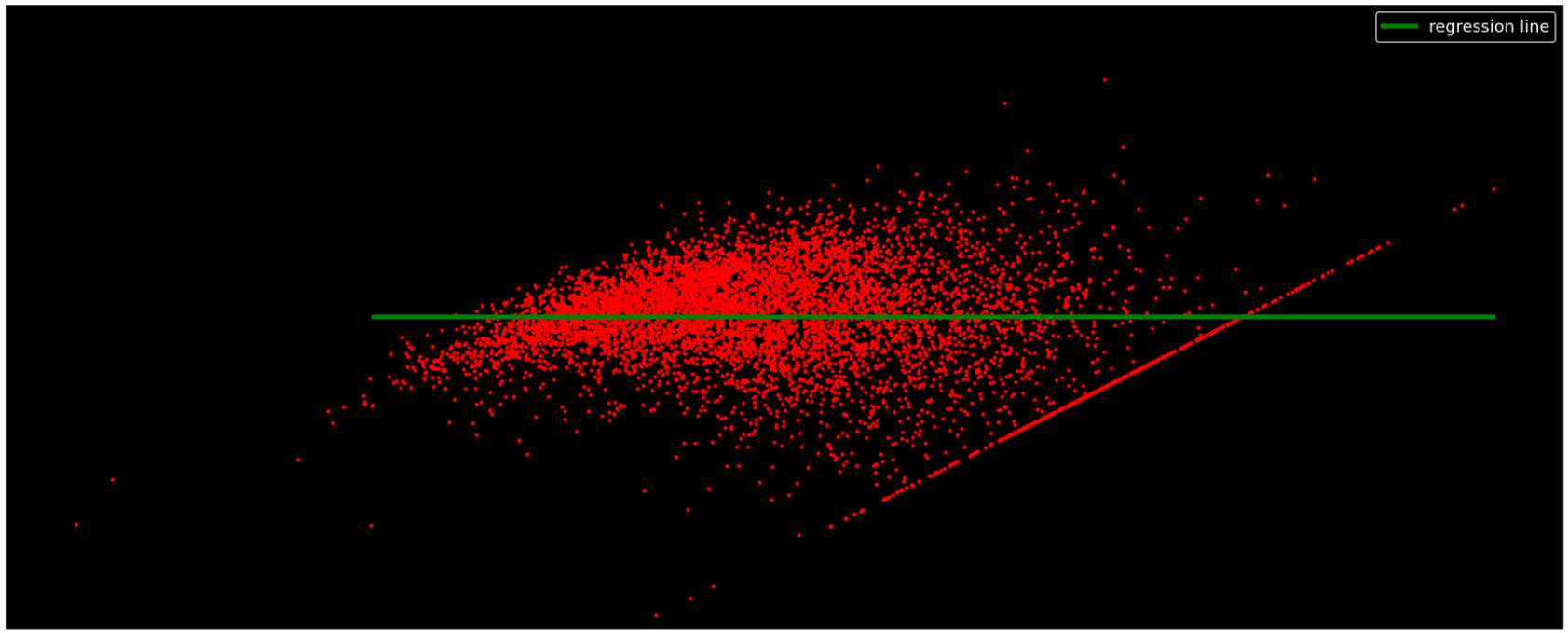
```
0.7344495220499551
```

```
residuals = predictions - y_test

residual_table = pd.DataFrame({'residuals':residuals,
                    'predictions':predictions})
residual_table = residual_table.sort_values( by = 'predictions')
```

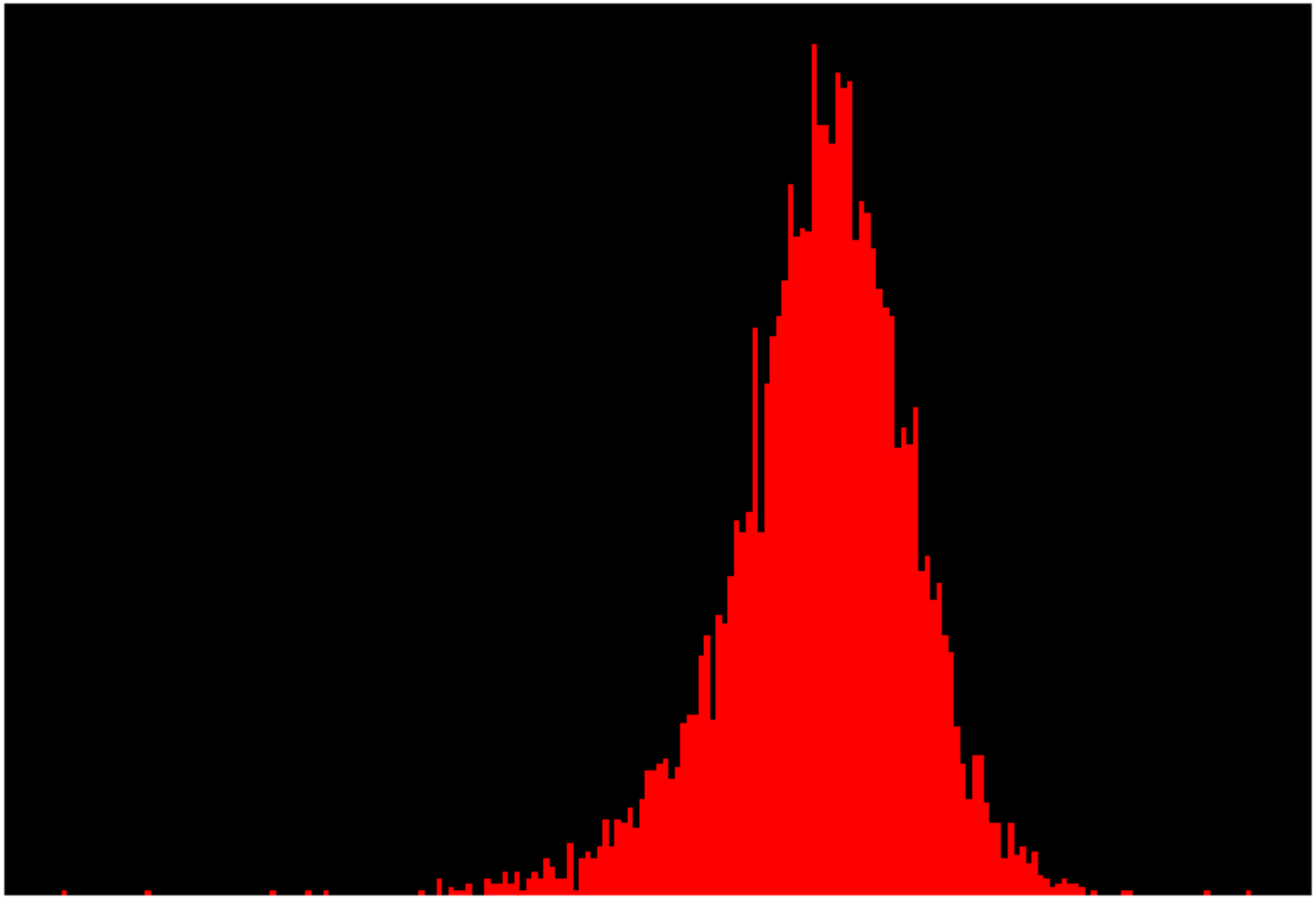
```
z = [i for i in range(int(residual_table['predictions'].max()))]
k = [0 for i in range(int(residual_table['predictions'].max()))]
```

```
plt.figure(dpi = 130, figsize = (17,7))

plt.scatter( residual_table['predictions'], residual_table['residuals'], color = 'red', s = 2)
plt.plot(z, k, color = 'green', linewidth = 3, label = 'regression line')
plt.ylim(-800000, 800000)
plt.xlabel('fitted points (ordered by predictions)')
plt.ylabel('residuals')
```

```
    plt.ylabel( residuals )
    plt.title('residual plot')
    plt.legend()
    plt.show()
```




```
    plt.figure(dpi = 100, figsize = (10,7))
    plt.hist(residual_table['residuals'], color = 'red', bins = 200)
    plt.xlabel('residuals')
    plt.ylabel('frequency')
    plt.title('distribution of residuals')
    plt.show()
```



```
coefficients_table = pd.DataFrame({'column': x_train.columns,
                                   'coefficients': lr.coef_})
coefficient_table = coefficients_table.sort_values(by = 'coefficients')
```

```
    plt.figure(figsize=(8, 6), dpi=120)
    x = coefficient_table['column']
    y = coefficient_table['coefficients']
    plt.barh( x, y)
    plt.xlabel( "Coefficients")
    plt.ylabel('Variables')
    plt.title('Normalized Coefficient plot')
    plt.show()
```