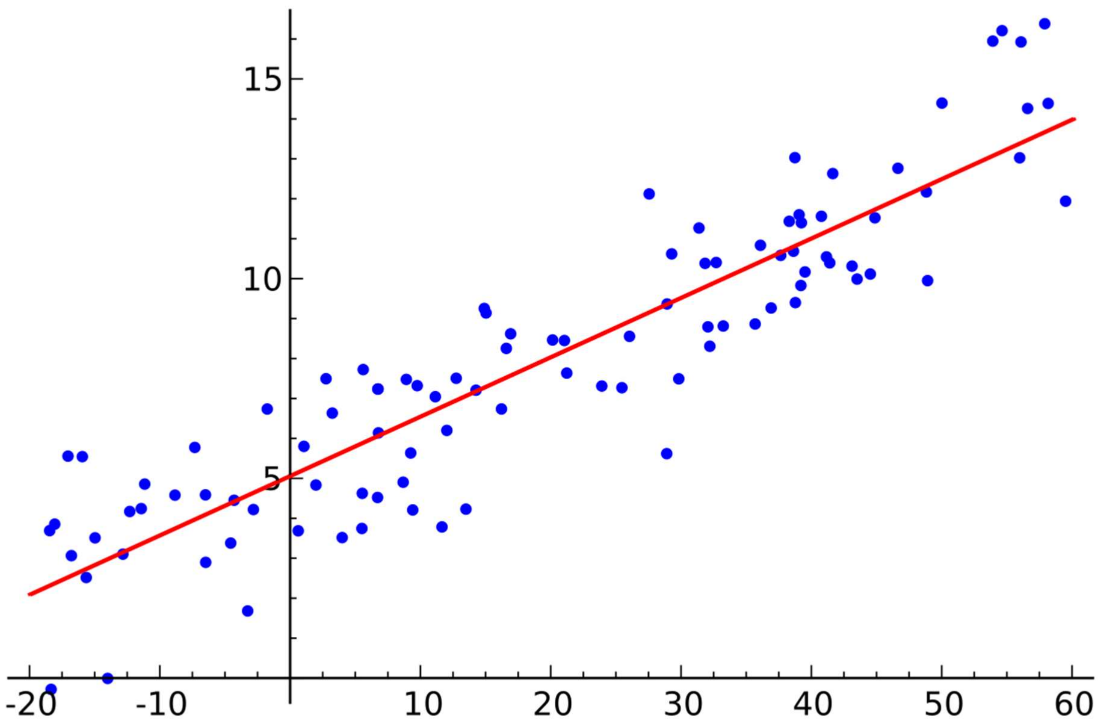
LINEAR REGRESSION MODEL



Predicting the Actual Sale Price of a House in North-Eastern areas of North America based on various factors

Date	24/12/2020
Name	Vinit Ravichandran Iyer
	Ashmitha Nagesh

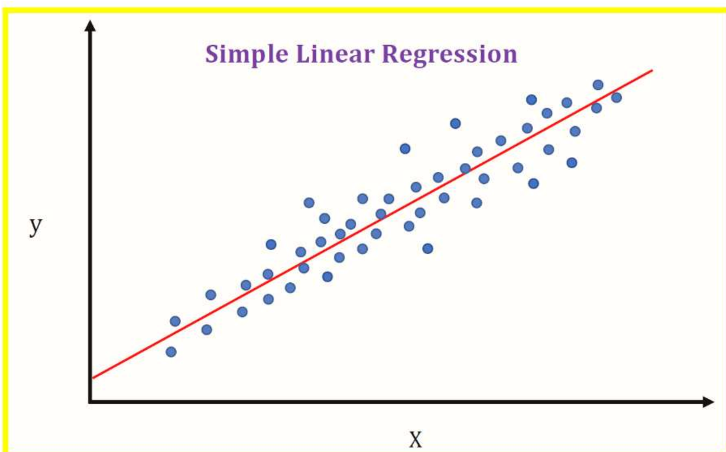
Content List

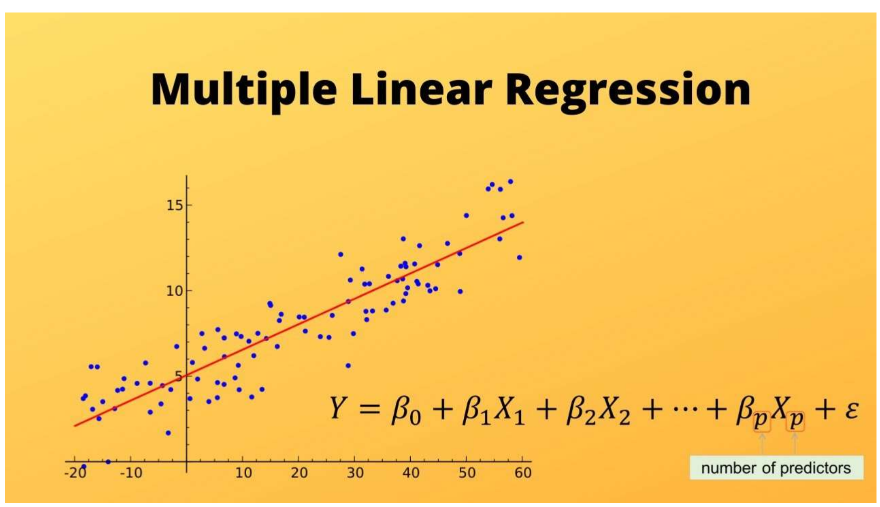
Introduction

Linear regression is a linear model, a model that assumes a linear relationship between the input variables (X) and the single output variable (Y) also known as the Target Variable. More specifically, that 'Y' can be calculated from a linear combination of the input variables (X). When there is a single input variable (X), the method is referred to as simple linear regression**.** When there are multiple input variables, it is known as multiple linear regression.

The following Linear Regression Model is created for the prediction of the actual Sale Price without any unnecessary monetary additions such as Brokerage, etc. The model utilizes various independent variables to identify the linear relationship between the independent and Target variable, in this case – Sale Price of the House.

A simple Linear Regression model consists of only a single variable on either axes whereas the Linear Regression Model created at the end of this project is based on multiple variables thus requiring various other methods to create a more suitable prediction. An example of each type of model is given below.





Source Code

1. Importing Libraries

Input	import numpy as np import matplotlib.pyplot as plt import seaborn as sea import pandas as pd

2. Importing Dataset

Input	data = pd.read_csv("Raw_Housing_Prices.csv") data.head()

Output

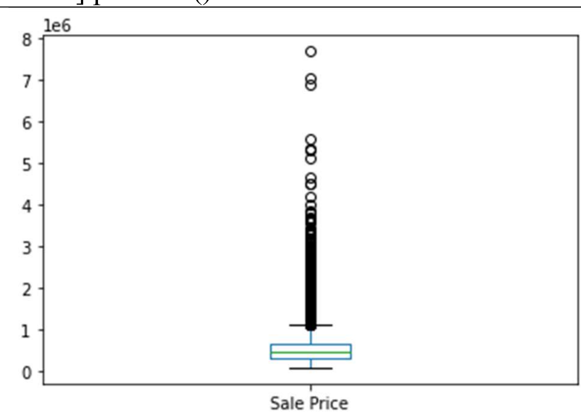
	ID	Date House was Sold	Sale Price	No of Bedrooms	No of Bathrooms	Flat Area (in Sqft)	Lot Area (in Sqft)	No of Floors	Waterfront View	No of Times Visited	...	Overall Grade	the House from Basement (in Sqft)	Basement Area (in Sqft)	of House (in Years)	Renovated Year
0	7129300520	14 October 2017	221900.0	3	1.00	1180.0	5650.0	1.0	No	None	...	7	1180.0	0	63	0
1	6414100192	14 December 2017	538000.0	3	2.25	2570.0	7242.0	2.0	No	None	...	7	2170.0	400	67	1991
2	5631500400	15 February 2016	180000.0	2	1.00	770.0	10000.0	1.0	No	None	...	6	770.0	0	85	0
3	2487200875	14 December 2017	604000.0	4	3.00	1960.0	5000.0	1.0	No	None	...	7	1050.0	910	53	0
4	1954400510	15 February 2016	510000.0	3	2.00	1680.0	8080.0	1.0	No	None	...	8	1680.0	0	31	0

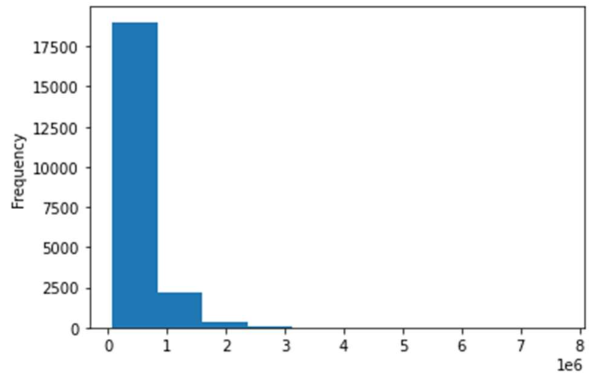
5 rows × 21 columns

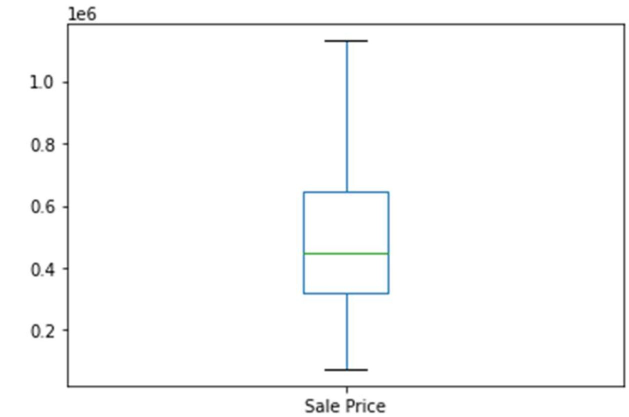
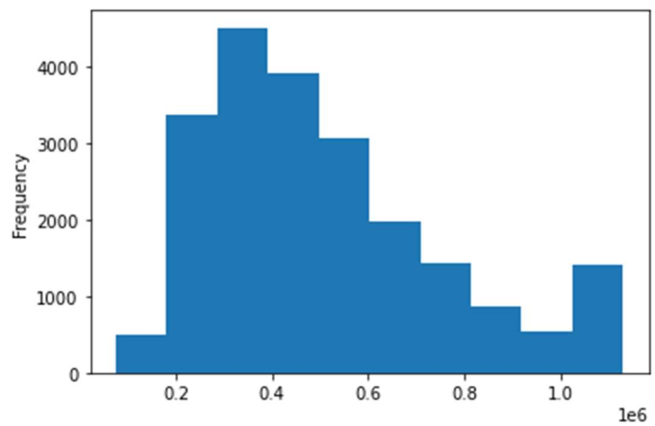
Input	data.info()
Output	<class 'pandas.core.frame.DataFrame'> RangeIndex: 21613 entries, 0 to 21612 Data columns (total 21 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 ID 21613 non-null int64 1 Date House was Sold 21613 non-null object 2 Sale Price 21609 non-null float64 3 No of Bedrooms 21613 non-null int64 4 No of Bathrooms 21609 non-null float64 5 Flat Area (in Sqft) 21604 non-null float64 6 Lot Area (in Sqft) 21604 non-null float64 7 No of Floors 21613 non-null float64

	8 Waterfront View 21613 non-null object 9 No of Times Visited 21613 non-null object 10 Condition of the House 21613 non-null object 11 Overall Grade 21613 non-null int64 12 Area of the House from Basement (in Sqft) 21610 non-null float64 13 Basement Area (in Sqft) 21613 non-null int64 14 Age of House (in Years) 21613 non-null int64 15 Renovated Year 21613 non-null int64 16 Zipcode 21612 non-null float64 17 Latitude 21612 non-null float64 18 Longitude 21612 non-null float64 19 Living Area after Renovation (in Sqft) 21612 non-null float64 20 Lot Area after Renovation (in Sqft) 21613 non-null int64 dtypes: float64(10), int64(7), object(4) memory usage: 3.5+ MB

3. Exploring Target Variable

Input	data["Sale Price"].describe()
Output	count 2.160900e+04 mean 5.401984e+05 std 3.673890e+05 min 7.500000e+04 25% 3.219500e+05 50% 4.500000e+05 75% 6.450000e+05 max 7.700000e+06 Name: Sale Price, dtype: float64
Input	data["Sale Price"].plot.box()
Output	

Input	data["Sale Price"].plot.hist()
Output	
Input	q1 = data["Sale Price"].quantile(0.25) q3 = data["Sale Price"].quantile(0.75) q1, q3
Output	(321950.0, 645000.0)
Input	iqr = q3-q1 iqr
Output	323050.0
Input	upper_limit = q3 + 1.5*iqr lower_limit = q1 - 1.5*iqr upper_limit, lower_limit
Output	(1129575.0, -162625.0)
Input	def new_limit(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(new_limit) data["Sale Price"].describe()
Output	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

<table>
<tr><td>Input</td><td>data["Sale Price"].plot.box()</td></tr>
<tr><td>Output</td><td>
</td></tr>
<tr><td>Input</td><td>data["Sale Price"].plot.hist()</td></tr>
<tr><td>Output</td><td>
</td></tr>
<tr><td>Input</td><td>data.isnull().sum()</td></tr>
<tr><td>Output</td><td>ID 0
Date House was Sold 0
Sale Price 4
No of Bedrooms 0
No of Bathrooms 4
Flat Area (in Sqft) 9
Lot Area (in Sqft) 9
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) 3
Basement Area (in Sqft) 0
Age of House (in Years) 0
Renovated Year 0
Zipcode 1
Latitude 1
Longitude 1
Living Area after Renovation (in Sqft) 1
Lot Area after Renovation (in Sqft) 0
dtype: int64</td></tr>
</table>

<table>
<tr><td>Input</td><td>data["Sale Price"].dropna(inplace = True)
data["Sale Price"].isnull().sum()</td></tr>
<tr><td>Output</td><td>0</td></tr>
<tr><td>Input</td><td>data.info()</td></tr>
<tr><td>Output</td><td><pre>
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column   Non-Nu
ll Count Dtype
--- ------   ------
-------- -----
 0   ID   21613
non-null int64
 1   Date House was Sold   21613
non-null object
 2   Sale Price   21609
non-null float64
 3   No of Bedrooms   21613
non-null int64
 4   No of Bathrooms   21609
non-null float64
 5   Flat Area (in Sqft)   21604
non-null float64
 6   Lot Area (in Sqft)   21604
non-null float64
 7   No of Floors   21613
non-null float64
 8   Waterfront View   21613
non-null object
 9   No of Times Visited   21613
non-null object
 10 Condition of the House   21613
non-null object
 11 Overall Grade   21613
non-null int64
 12 Area of the House from Basement (in Sqft) 21610
non-null float64
 13 Basement Area (in Sqft)   21613
non-null int64
 14 Age of House (in Years)   21613
non-null int64
 15 Renovated Year   21613
non-null int64
 16 Zipcode   21612
non-null float64
 17 Latitude   21612
non-null float64
 18 Longitude   21612
non-null float64
 19 Living Area after Renovation (in Sqft)   21612
non-null float64
 20 Lot Area after Renovation (in Sqft)   21613
non-null int64
dtypes: float64(10), int64(7), object(4)
memory usage: 3.5+ MB
</pre></td></tr>
</table>

4. Exploring Other Variables

<table>
<tr><td>Input</td><td>

```
#Float 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)"]

from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values = np.nan, strategy = "median")
data[numerical_columns] = imputer.fit_transform(data[numerical_columns])

data.info()
```

</td></tr>
<tr><td>Output</td><td>

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column                                     Non-Nu
ll Count  Dtype
---  ------                                     ------
--------  -----
 0   ID                                         21613
non-null  int64
 1   Date House was Sold                        21613
non-null  object
 2   Sale Price                                 21609
non-null  float64
 3   No of Bedrooms                             21613
non-null  int64
 4   No of Bathrooms                            21613
non-null  float64
 5   Flat Area (in Sqft)                        21613
non-null  float64
 6   Lot Area (in Sqft)                         21613
non-null  float64
 7   No of Floors                               21613
non-null  float64
 8   Waterfront View                            21613
non-null  object
 9   No of Times Visited                        21613
non-null  object
 10  Condition of the House                     21613
non-null  object
 11  Overall Grade                              21613
non-null  int64
 12  Area of the House from Basement (in Sqft)  21613
non-null  float64
 13  Basement Area (in Sqft)                    21613
non-null  int64
 14  Age of House (in Years)                    21613
non-null  int64
 15  Renovated Year                             21613
non-null  int64
 16  Zipcode                                    21612
non-null  float64
```

</td></tr>
</table>

	`` 17 Latitude 21613 non-null float64 18 Longitude 21613 non-null float64 19 Living Area after Renovation (in Sqft) 21613 non-null float64 20 Lot Area after Renovation (in Sqft) 21613 non-null int64 dtypes: float64(10), int64(7), object(4) memory usage: 3.5+ MB ```

5. Transformation of Variables – Zipcode

Input	imputer = SimpleImputer(missing_values = np.nan, strategy = "most_frequent") data["Zipcode"] = imputer.fit_transform(data["Zipcode"].values.reshape(-1,1)) data["Zipcode"].shape
Output	(21613,)
Input	column = data["Zipcode"].values.reshape(-1,1) column.shape
Output	(21613, 1)
Input	imputer = SimpleImputer(missing_values = np.nan, strategy = "most_frequent") data["Zipcode"] = imputer.fit_transform(column) data.info()
Output	<class 'pandas.core.frame.DataFrame'> RangeIndex: 21613 entries, 0 to 21612 Data columns (total 21 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 ID 21613 non-null int64 1 Date House was Sold 21613 non-null object 2 Sale Price 21609 non-null float64 3 No of Bedrooms 21613 non-null int64 4 No of Bathrooms 21613 non-null float64 5 Flat Area (in Sqft) 21613 non-null float64 6 Lot Area (in Sqft) 21613 non-null float64 7 No of Floors 21613 non-null float64 8 Waterfront View 21613 non-null object 9 No of Times Visited 21613 non-null object

	` 10  Condition of the House                     21613` `non-null  object` ` 11  Overall Grade                              21613` `non-null  int64` ` 12  Area of the House from Basement (in Sqft)  21613` `non-null  float64` ` 13  Basement Area (in Sqft)                    21613` `non-null  int64` ` 14  Age of House (in Years)                    21613` `non-null  int64` ` 15  Renovated Year                             21613` `non-null  int64` ` 16  Zipcode                                    21613` `non-null  float64` ` 17  Latitude                                   21613` `non-null  float64` ` 18  Longitude                                  21613` `non-null  float64` ` 19  Living Area after Renovation (in Sqft)     21613` `non-null  float64` ` 20  Lot Area after Renovation (in Sqft)        21613` `non-null  int64` `dtypes: float64(10), int64(7), object(4)` `memory usage: 3.5+ MB`

6. Transformation of Variables – others

Input	data["No of Times Visited"].unique()
Output	`array(['None', 'Thrice', 'Four', 'Twice', 'Once'], dtype=object)`
Input	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()
Output	`array(['0', '3', '4', '2', '1'], dtype=object)`

7. Creating more valuable columns

Input	data["Ever Renovated"] = np.where(data["Renovated Year"] == 0, "No", "Yes") data.head(2)

	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	...	Area of the House from Basement (in Sqft)	Basement Area (in Sqft)	Age of House (in Years)	Renovated Year	Zipcode
0	7129300520	14 October 2017	221900.0	3	1.00	1180.0	5650.0	1.0	No	0	...	1180.0	0	63	0	98178.0
1	6414100192	14 December 2017	538000.0	3	2.25	2570.0	7242.0	2.0	No	0	...	2170.0	400	67	1991	98125.0

2 rows × 22 columns

Input	data['Purchase Year'] = pd.DatetimeIndex(data['Date House was Sold']).year data["Year since Renovation"] = np.where(data["Ever Renovated"] == "Yes", abs(data["Purchase Year"] - data["Renovated Year"]), 0) data.head(2)

	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	...	Age of House (in Years)	Renovated Year	Zipcode	Latitude	Longitude	L R
0	7129300520	14 October 2017	221900.0	3	1.00	1180.0	5650.0	1.0	No	0	...	63	0	98178.0	47.5112	-122.257	
1	6414100192	14 December 2017	538000.0	3	2.25	2570.0	7242.0	2.0	No	0	...	67	1991	98125.0	47.7210	-122.319	

2 rows × 24 columns

Input	data.drop(columns = ["Purchase Year", "Date House was Sold", "Renovated Year"], inplace = True) data.head(2)

```
1 data.head(2)
```

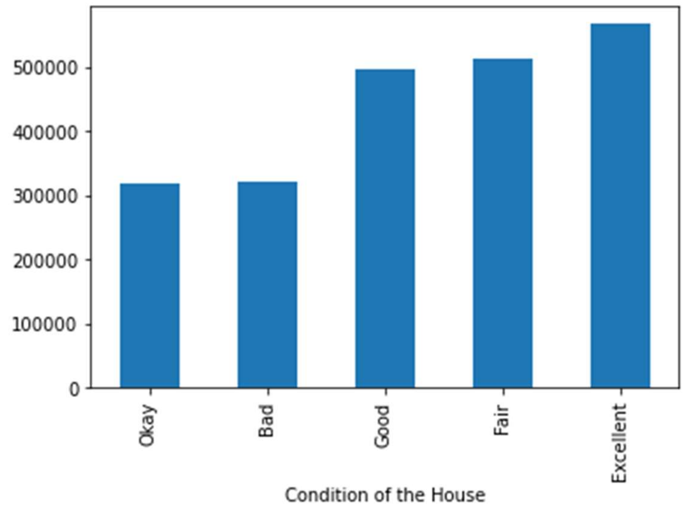
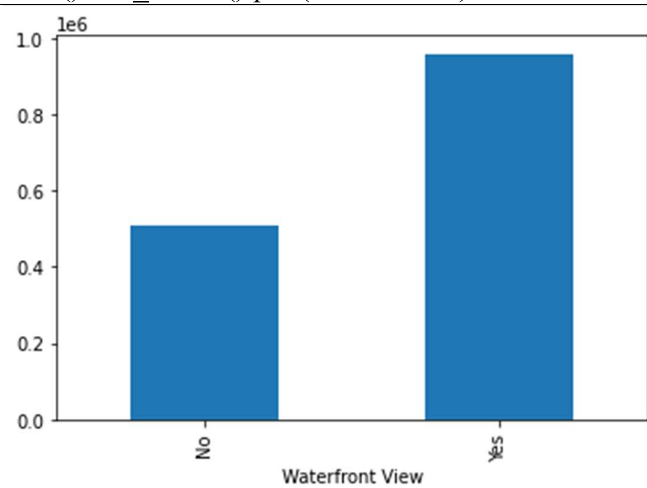
	ID	Sale Price	No of Bedrooms	No of Bathrooms	Flat Area (in Sqft)	Lot Area (in Sqft)	No of Floors	Waterfront View	No of Times Visited	Condition of the House	...	Area of the House from Basement (in Sqft)	Basement Area (in Sqft)	Age of House (in Years)	Zipcode	Latitude	L
0	7129300520	221900.0	3	1.00	1180.0	5650.0	1.0	No	0	Fair	...	1180.0	0	63	98178.0	47.5112	
1	6414100192	538000.0	3	2.25	2570.0	7242.0	2.0	No	0	Fair	...	2170.0	400	67	98125.0	47.7210	

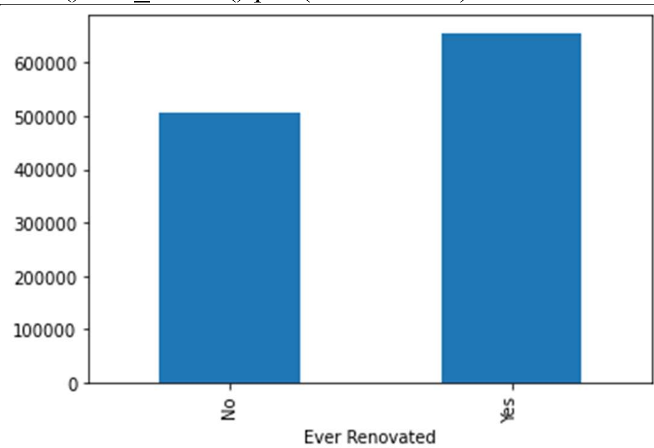
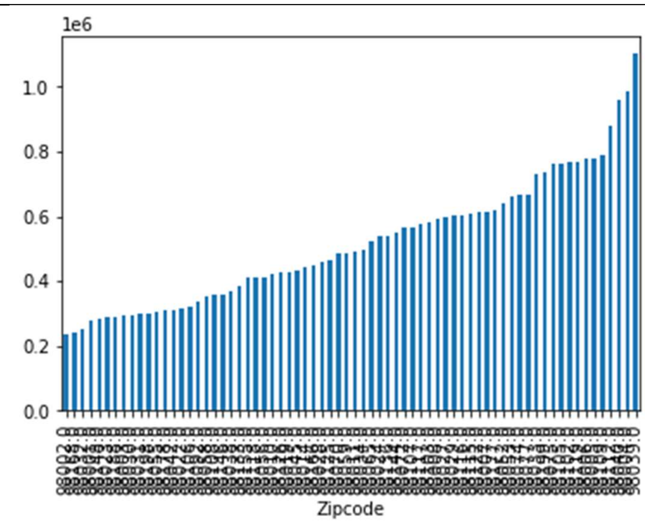
2 rows × 21 columns

8. Grouping the variables

Input	data.drop(columns = "ID", inplace = True) data.head(2)

	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 of the House from Basement (in Sqft)	Basement Area (in Sqft)	Age of House (in Years)	Zipcode	Latitude	Longitude
0	221900.0	3	1.00	1180.0	5650.0	1.0	No	0	Fair	7	1180.0	0	63	98178.0	47.5112	-122.257
1	538000.0	3	2.25	2570.0	7242.0	2.0	No	0	Fair	7	2170.0	400	67	98125.0	47.7210	-122.319

Input	data["Condition of the House"].value_counts()
Output	`Fair 14031` `Good 5679` `Excellent 1701` `Okay 172` `Bad 30` `Name: Condition of the House, dtype: int64`
Input	data.groupby("Condition of the House")["Sale Price"].mean().sort_values().plot(kind = "bar")
Output	
Input	data.groupby("Waterfront View")["Sale Price"].mean().sort_values().plot(kind = "bar")
Output	

Input	data.groupby("Ever Renovated")["Sale Price"].mean().sort_values().plot(kind = "bar")
Output	
Input	data.groupby("Zipcode")["Sale Price"].mean().sort_values().plot(kind = "bar")
Output	

9. Splitting the Dataset into Independent and Target Variables

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

10. Transformation of Variables

<table>
<tr><td>Input</td><td><pre><code>def distribution(data ,var):
  plt.figure(figsize = (len(var)*10,10), dpi = 200)
  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)', 'Year since Renovation']

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

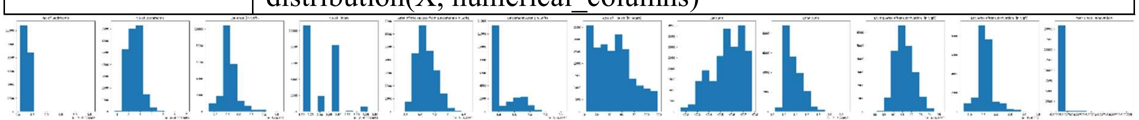
distribution(X, numerical_columns)</code></pre></td></tr>
<tr><td>Input</td><td><pre><code>def right_skew(x):
  return np.log(abs(x+500))

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)', 'Year 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)</code></pre></td></tr>
</table>



11. Scaling the Dataset

Input	X.head()

of s	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)	Age of House (in Years)	Zipcode	Latitude	Longitude	Living Area after Renovation (in Sqft)	L Renc (i
35	6.226985	1180.0	6.226995	6.226985	No	0	Fair	7	6.226990	6.226985	63	98178.0	47.5112	6.226984	6.226990	6.
35	6.226985	2570.0	6.226996	6.226985	No	0	Fair	7	6.226992	6.226987	67	98125.0	47.7210	6.226984	6.226991	6.
35	6.226985	770.0	6.226997	6.226985	No	0	Fair	6	6.226989	6.226985	85	98028.0	47.7379	6.226984	6.226992	6.
35	6.226985	1960.0	6.226994	6.226985	No	0	Excellent	7	6.226990	6.226989	53	98136.0	47.5208	6.226984	6.226990	6.
35	6.226985	1680.0	6.226996	6.226985	No	0	Fair	8	6.226991	6.226985	31	98074.0	47.6168	6.226984	6.226991	6.

Input	Y.head()
Output	`0     221900.0` `1     538000.0` `2     180000.0` `3     604000.0` `4     510000.0` `Name: Sale Price, dtype: float64`
Input	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()

a t)	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)	Age of House (in Years)	Zipcode	Latitude	Longitude	Living Area after Renovation (in Sqft)	Lot Area after Renovation (in Sqft)	Ever Renovated	Year since Renovation
5	6.226985	0	0	3	7	6.226990	6.226985	63	98178.0	47.5112	6.226984	6.226990	6.226995	0	6.226961
6	6.226985	0	0	3	7	6.226992	6.226987	67	98125.0	47.7210	6.226984	6.226991	6.226996	1	6.227061
7	6.226985	0	0	3	6	6.226989	6.226985	85	98028.0	47.7379	6.226984	6.226992	6.226996	0	6.226961
4	6.226985	0	0	5	7	6.226990	6.226989	53	98136.0	47.5208	6.226984	6.226990	6.226994	0	6.226961
6	6.226985	0	0	3	8	6.226991	6.226985	31	98074.0	47.6168	6.226984	6.226991	6.226996	0	6.226961

Input	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)	Age of House (in Years)	Zipcode	Latitude	Lc
0	-0.398646	-1.448933	-0.979905	-0.411841	-0.915605	-0.087181	-0.30579	-0.629203	-0.563993	-0.767575	-0.726430	0.544734	1.870094	-0.352576	-(
1	-0.398646	0.176497	0.533718	-0.138804	0.937194	-0.087181	-0.30579	-0.629203	-0.563993	0.642316	0.539016	0.680915	0.879534	1.161645	-(
2	-1.477795	-1.448933	-1.426369	0.222411	-0.915605	-0.087181	-0.30579	-0.629203	-1.468566	-1.619630	-0.726430	1.293731	-0.933379	1.283619	-(
3	0.678355	1.149811	-0.130534	-0.544371	-0.915605	-0.087181	-0.30579	2.444136	-0.563993	-1.012806	1.504571	0.204281	1.085122	-0.283288	-
4	-0.398646	-0.148264	-0.435436	-0.016950	-0.915605	-0.087181	-0.30579	-0.629203	0.340581	0.025445	-0.726430	-0.544715	-0.073647	0.409587	

12. Multicollinearity Check and removal

Input	X.corr()

	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)	Age of House (in Years)	Zipcode	L
No of Bedrooms	1.000000	0.516646	0.577470	0.175715	0.175996	-0.006617	0.079649	0.028514	0.349935	0.509519	0.276721	-0.154614	-0.153164	-0.
No of Bathrooms	0.516646	1.000000	0.754414	0.104886	0.500980	0.063683	0.187657	-0.124874	0.635778	0.696037	0.253984	-0.506206	-0.204098	0.
Flat Area (in Sqft)	0.577470	0.754414	1.000000	0.341686	0.354268	0.103841	0.284678	-0.058922	0.705725	0.853616	0.373178	-0.318146	-0.199380	0.
Lot Area (in Sqft)	0.175715	0.104886	0.341686	1.000000	-0.218973	0.074354	0.121725	0.066323	0.165721	0.319775	0.056326	-0.005815	-0.279421	-0.
No of Floors	0.175996	0.500980	0.354268	-0.218973	1.000000	0.023721	0.029503	-0.263676	0.461442	0.548423	-0.266623	-0.489232	-0.059289	0.
Waterfront View	-0.006617	0.063683	0.103841	0.074354	0.023721	1.000000	0.401856	0.016650	0.070332	0.063276	0.063249	0.026149	0.030286	-0.
No of Times Visited	0.079649	0.187657	0.284678	0.121725	0.029503	0.401856	1.000000	0.045978	0.223661	0.161089	0.249394	0.053395	0.084830	0.
Condition of the House	0.028514	-0.124874	-0.058922	0.066323	-0.263676	0.016650	0.045978	1.000000	-0.143747	-0.153567	0.176036	0.361383	0.003076	-0.
Overall Grade	0.349935	0.635778	0.705725	0.165721	0.461442	0.070332	0.223661	-0.143747	1.000000	0.723787	0.116024	-0.456711	-0.185844	0
Area of the House from Basement (in Sqft)	0.509519	0.696037	0.853616	0.319775	0.548423	0.063276	0.161089	-0.153567	0.723787	1.000000	-0.111373	-0.448716	-0.285312	-0.
Basement Area (in Sqft)	0.276721	0.253984	0.373178	0.056326	-0.266623	0.063249	0.249394	0.176036	0.116024	-0.111373	1.000000	0.153959	0.103575	0.

13. Calculating Variance Inflation Factor (VIF)

Input	from statsmodels.stats.outliers_influence import variance_inflation_factor vif_data = X[:] VIF = pd.Series([variance_inflation_factor(vif_data.values, i) for i in range(vif_data.shape[1])], index = vif_data.columns) VIF
Output	No of Bedrooms 1.736922 No of Bathrooms 3.424525 Flat Area (in Sqft) 21.438595 Lot Area (in Sqft) 6.854177 No of Floors 2.390210 Waterfront View 1.211023 No of Times Visited 1.415563 Condition of the House 1.260555 Overall Grade 2.905740 Area of the House from Basement (in Sqft) 23.213285 Basement Area (in Sqft) 6.541646 Age of House (in Years) 2.458385 Zipcode 1.668851 Latitude 1.191524 Longitude 1.880325 Living Area after Renovation (in Sqft) 2.916939 Lot Area after Renovation (in Sqft) 6.610205 Ever Renovated 3.022932 Year since Renovation 2.872220 dtype: float64
Input	def Multicollinearity_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(10): vif_data = Multicollinearity_remover(vif_data) vif_data.head()
Output	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 No Multicollinearity present anymore No Multicollinearity present anymore No Multicollinearity present anymore No Multicollinearity present anymore No Multicollinearity present anymore

	No of Bedrooms	No of Bathrooms	No of Floors	Waterfront View	No of Times Visited	Condition of the House	Overall Grade	Basement Area (in Sqft)	Age of House (in Years)	Zipcode	Latitude	Longitude	Living Area after Renovation (in Sqft)	Lot Area after Renovation (in Sqft)
0	-0.398646	-1.448933	-0.915605	-0.087181	-0.30579	-0.629203	-0.563993	-0.726430	0.544734	1.870094	-0.352576	-0.306108	-1.027661	-0.416286
1	-0.398646	0.176497	0.937194	-0.087181	-0.30579	-0.629203	-0.563993	0.539016	0.680915	0.879534	1.161645	-0.746519	-0.355795	-0.047629
2	-1.477795	-1.448933	-0.915605	-0.087181	-0.30579	-0.629203	-1.468566	-0.726430	1.293731	-0.933379	1.283619	-0.135646	1.130676	0.019007
3	0.678355	1.149811	-0.915605	-0.087181	-0.30579	2.444136	-0.563993	1.504571	0.204281	1.085122	-0.283288	-1.272267	-0.985943	-0.563303
4	-0.398646	-0.148264	-0.915605	-0.087181	-0.30579	-0.629203	0.340581	-0.726430	-0.544715	-0.073647	0.409587	1.199268	-0.166751	-0.069793

<table>
<tr><td>Input</td><td>

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

</td></tr>
<tr><td>Output</td><td>

```
(No of Bedrooms                            1.498201
 No of Bathrooms                           2.950074
 No of Floors                              2.186151
 Waterfront View                           1.209183
 No of Times Visited                       1.410552
 Condition of the House                    1.253804
 Overall Grade                             2.541289
 Basement Area (in Sqft)                   1.639832
 Age of House (in Years)                   2.392458
 Zipcode                                   1.666022
 Latitude                                  1.183418
 Longitude                                 1.857959
 Living Area after Renovation (in Sqft)    2.503690
 Lot Area after Renovation (in Sqft)       1.553976
 Ever Renovated                            3.017750
 Year since Renovation                     2.868646
 dtype: float64,
 16)
```

</td></tr>
<tr><td>Input</td><td>

```
X = vif_data[:]

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

</td></tr>
</table>

14. Splitting Dataset into Training and Test dataset

<table>
<tr><td>Input</td><td>

```
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
```

</td></tr>
<tr><td>Output</td><td>

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

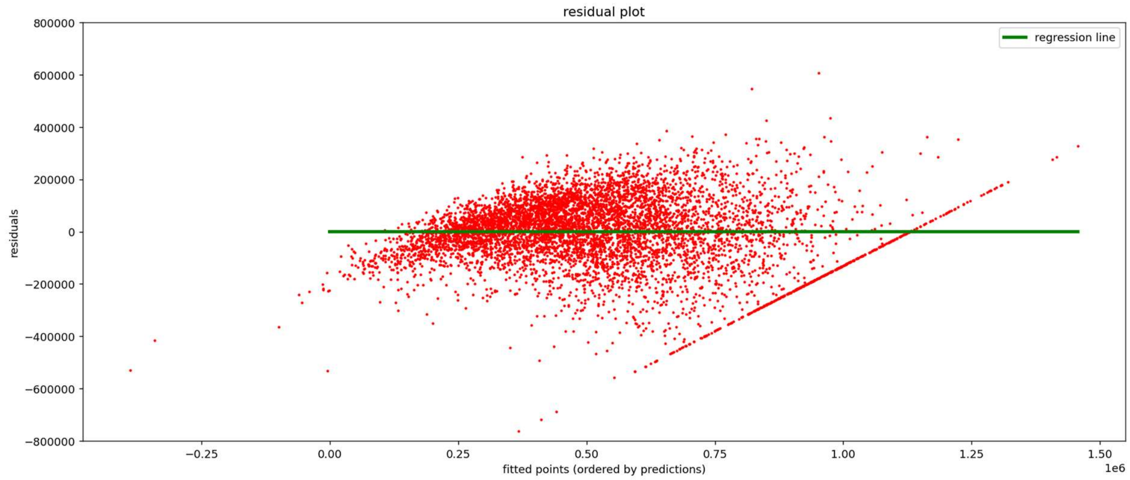
</td></tr>
</table>

15. Training the Model

Input	from sklearn.linear_model import LinearRegression lr = LinearRegression(normalize = True) lr.fit(x_train, y_train)
Output	LinearRegression(normalize=True)
Input	lr.coef_
Output	array([1586.16613781, 42606.86490238, 23305.62441544, 9989.75886541, 30485.29883092, 16059.34035324, 108947.72958151, 11325.73956762, 65052.29861961, -15609.10786456, 75616.69770673, -7742.01109724, 54276.77846551, 2006.20731051, 16443.61945338, -11319.94397008])
Input	predictions = lr.predict(x_test) lr.score(x_test, y_test)
Output	0.7344347887730929

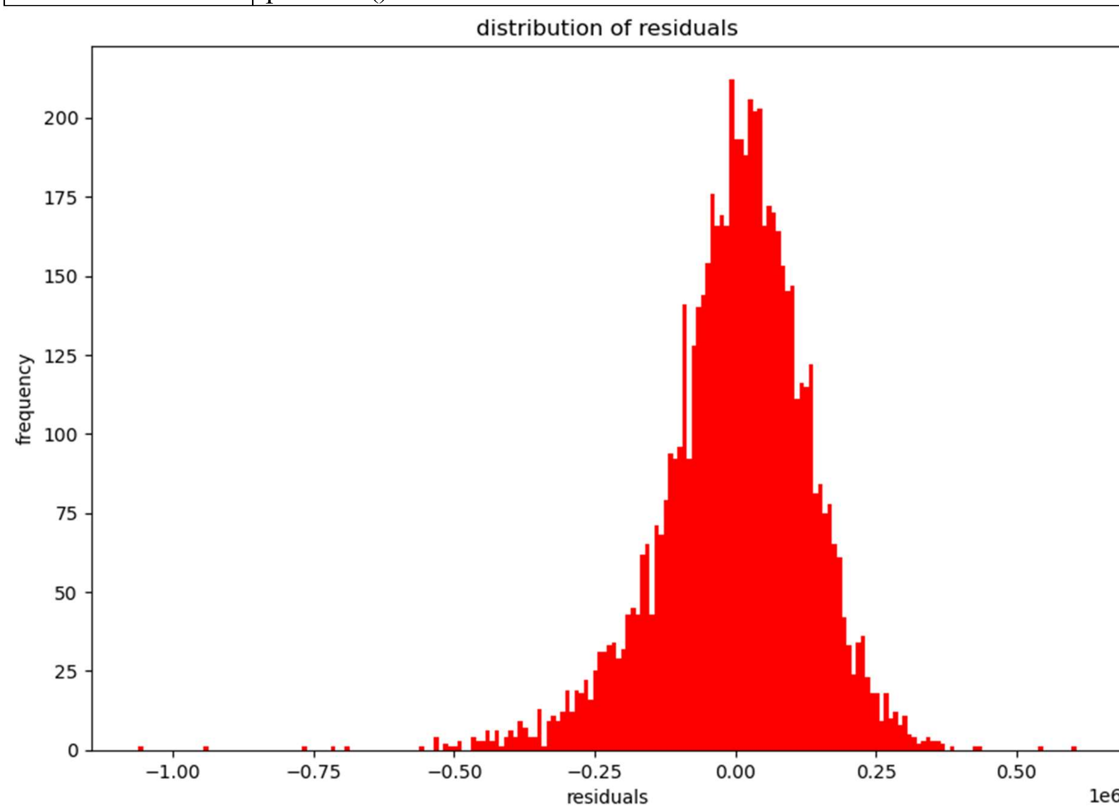
16. Plotting Residuals

Input	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.title('residual plot') plt.legend() plt.show()



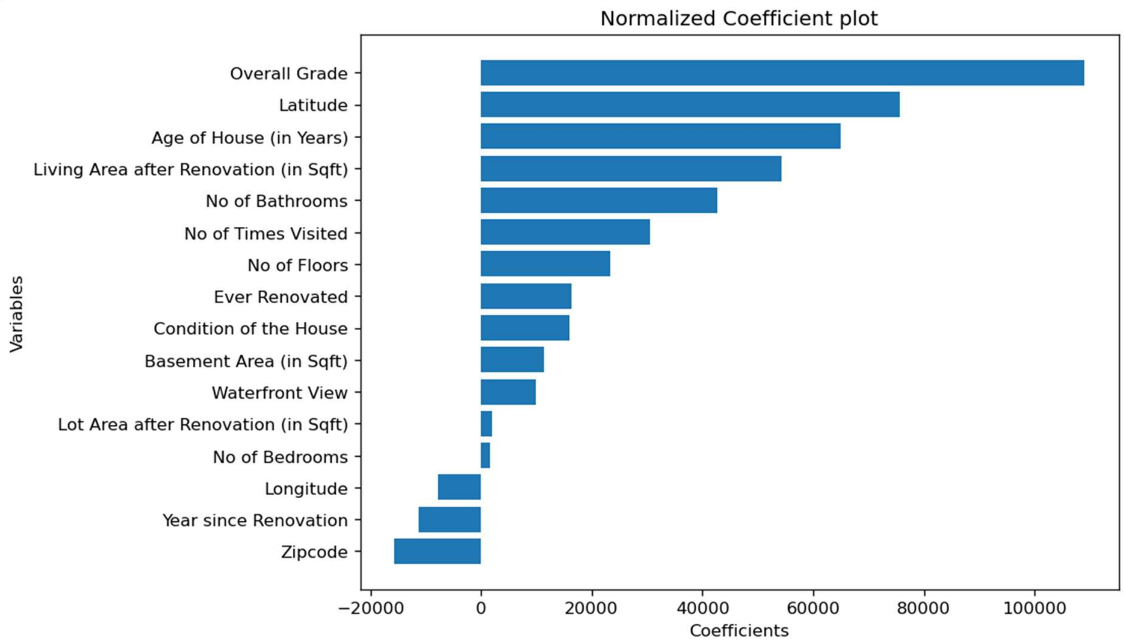
17. Plotting Error Distribution

Input	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()

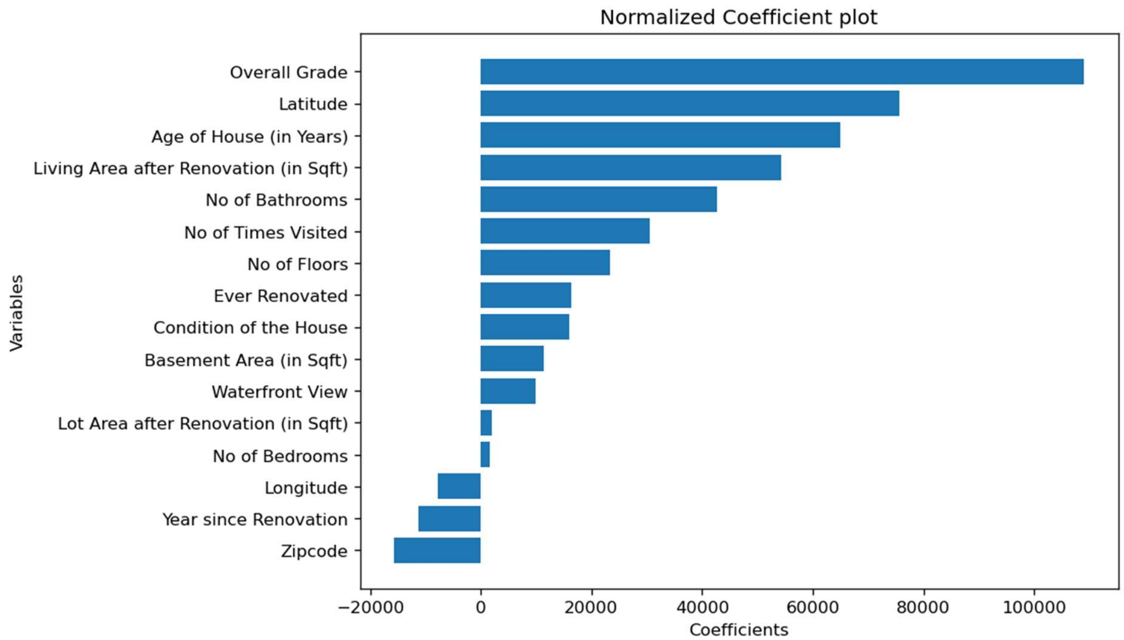


18. Model Coefficients

Input	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()



Coefficient Output



Inference

The Bar Graph provides the relationship between the multiple variables and the target variable, Sale Price of the House. As seen, multiple factors affect the sale price with the Overall Grade being the one having the most positive impact and zip-code having the most negative impact on the Sale Price of the House.

The No of bedrooms and Lot area after Renovation actually has minimum impact on the sale price. Another thing that can be inferred is that the Latitude plays an important role in the sale price whereas the Longitude has a negative impact albeit little on the sale price.

Accordingly, the more older the house is the more the cost and renovated houses are priced more according to the Living area after renovation. Ironically the number of Bedrooms have the most minimum effect on the sale price as compared to the number of Bathrooms or even number of Floors.

Thus based on the coefficients obtained from the Linear Regression Model, one would be able to predict the Sale Price of a house in the area the primary dataset has been obtained from.