Data Preprocessing

```
import pandas as pd
import  numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('/content/drive/MyDrive/dataset')
df.head()
```

	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	Lattitud
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4	5	...	1921	0	122003	52.864
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0	5	...	1909	0	122004	52.887
2	6762810998	42491	5	2.75	2910	9480	1.5	0	0	3	...	1939	0	122004	52.885
3	6762812605	42491	4	2.50	3310	42998	2.0	0	0	3	...	2001	0	122005	52.953
4	6762812919	42491	3	2.00	2710	4500	1.5	0	0	4	...	1929	0	122006	52.904

Saved successfully!

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df.tail()
```

	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	Latt
14615	6762830250	42734	2	1.5	1556	20000	1.0	0	0	4	...	1957	0	122066	52
14616	6762830339	42734	3	2.0	1680	7000	1.5	0	0	4	...	1968	0	122072	52
14617	6762830618	42734	2	1.0	1070	6120	1.0	0	0	3	...	1962	0	122056	52
14618	6762830709	42734	4	1.0	1030	6621	1.0	0	0	4	...	1955	0	122042	52
14619	6762831463	42734	3	1.0	900	4770	1.0	0	0	3	...	1969	2009	122018	52

5 rows × 23 columns

```
df.shape
```

```
(14620, 23)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 23 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              14620 non-null  int64  
 1   Date                            14620 non-null  int64  
 2   number of bedrooms              14620 non-null  int64  
 3   number of bathrooms             14620 non-null  float64
 4   living area                     14620 non-null  int64  
 5   lot area                        14620 non-null  int64  
```

```
 6   number of floors                      14620 non-null  float64
 7   waterfront present                    14620 non-null  int64
 8   number of views                       14620 non-null  int64
 9   condition of the house                14620 non-null  int64
 10  grade of the house                    14620 non-null  int64
 11  Area of the house(excluding basement) 14620 non-null  int64
 12  Area of the basement                  14620 non-null  int64
 13  Built Year                            14620 non-null  int64
 14  Renovation Year                       14620 non-null  int64
 15  Postal Code                           14620 non-null  int64
 16  Lattitude                             14620 non-null  float64
 17  Longitude                             14620 non-null  float64
 18  living_area_renov                     14620 non-null  int64
 19  lot_area_renov                        14620 non-null  int64
 20  Number of schools nearby              14620 non-null  int64
 21  Distance from the airport             14620 non-null  int64
 22  Price                                 14620 non-null  int64
dtypes: float64(4), int64(19)
memory usage: 2.6 MB
```

```
df.isnull().any()
```

```
id                                    False
Date                                  False
number of bedrooms                    False
number of bathrooms                   False
living area                           False
lot area                              False
                                      False
                                      False
                                      False
condition of the house                False
grade of the house                    False
Area of the house(excluding basement) False
Area of the basement                  False
Built Year                            False
Renovation Year                       False
Postal Code                           False
Lattitude                             False
Longitude                             False
living_area_renov                     False
lot_area_renov                        False
Number of schools nearby              False
Distance from the airport             False
Price                                 False
dtype: bool
```

Saved successfully!

Missing values

```
# if we have null values in the categorical columns we have to replace it with mode



## For the numerical column -null values replacement
## df['lotarea'].fillna['lotarea'].median(),inplace=True
##or
## df['lotarea'].fillna['lotarea'].mean(),inplace=True()


## For the categorical column-null values replacement

## df['lotarea'].fillna(df['lotarea'].mode(),inplace=True)
```

Describbstive statistic

```
df.head()
```

	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	Lattitud
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4	5	...	1921	0	122003	52.864
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0	5	...	1909	0	122004	52.887

```
df.Date.unique()
```

```
array([42491, 42492, 42493, 42494, 42495, 42496, 42497, 42498, 42499,
       42500, 42501, 42502, 42503, 42504, 42505, 42506, 42507, 42508,
       42509, 42510, 42511, 42512, 42513, 42514, 42515, 42516, 42517,
       42518, 42519, 42520, 42521, 42522, 42523, 42524, 42525, 42526,
       42527, 42528, 42529, 42530, 42531, 42532, 42533, 42534, 42535,
       42536, 42537, 42538, 42539, 42540, 42541, 42542, 42543, 42544,
       42545, 42546, 42547, 42548, 42549, 42550, 42551, 42552, 42553,
       42554, 42555, 42556, 42557, 42558, 42559, 42560, 42561, 42562,
       42563, 42564, 42565, 42566, 42567, 42568, 42569, 42570, 42571,
       42572, 42573, 42574, 42575, 42576, 42577, 42578, 42579, 42580,
       42581, 42582, 42583, 42584, 42585, 42586, 42587, 42588, 42589,
       42590, 42591, 42592, 42593, 42594, 42595, 42596, 42597, 42598,
       42599, 42600, 42601, 42602, 42603, 42604, 42605, 42606, 42607,
       42608, 42609, 42610, 42611, 42612, 42613, 42614, 42615, 42616,
       42617, 42618, 42619, 42620, 42621, 42622, 42623, 42624, 42625,
       42626, 42627, 42628, 42629, 42630, 42631, 42632, 42633, 42634,
       42635, 42636, 42637, 42638, 42639, 42640, 42641, 42642, 42643,
       42644, 42645, 42646, 42647, 42648, 42649, 42650, 42651, 42652,
       42680, 42681, 42682, 42683, 42684, 42685, 42686, 42687, 42688,
       42689, 42690, 42691, 42692, 42693, 42694, 42695, 42696, 42697,
       42698, 42699, 42701, 42702, 42703, 42704, 42705, 42706, 42707,
       42708, 42709, 42710, 42711, 42712, 42713, 42714, 42715, 42716,
       42717, 42718, 42719, 42720, 42721, 42722, 42723, 42724, 42725,
       42726, 42727, 42729, 42730, 42732, 42733, 42734])
```

Saved successfully!

```
df.id.value_counts()
```

```
6762810145    1
6762812541    1
6762813736    1
6762813785    1
6762814009    1
             ..
6762812356    1
6762812410    1
6762812471    1
6762812563    1
6762831463    1
Name: id, Length: 14620, dtype: int64
```

```
df.Longitude.value_counts()
```

```
-114.552    73
-114.562    73
-114.490    72
-114.480    69
-114.553    68
            ..
-114.036     1
-114.096     1
-113.994     1
-114.111     1
-113.891     1
Name: Longitude, Length: 716, dtype: int64
```

```
df.Lattitude.value_counts()
```

```
52.7745    13
52.8947    13
52.9255    13
52.9146    13
52.8834    13
           ..
52.8969     1
52.6834     1
52.5337     1
52.6155     1
```

```
52.5338     1
Name: Lattitude, Length: 4662, dtype: int64
```

```
df.corr()
```

	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
id	1.000000	0.045966	-0.329034	-0.516909	-0.648127	-0.100269	-0.312305	-0.112937	-0.293004	-0.045061	...	-0.068645
Date	0.045966	1.000000	-0.015663	-0.026485	-0.021958	0.004392	-0.010335	0.012006	-0.004782	-0.027402	...	-0.005869
number of bedrooms	-0.329034	-0.015663	1.000000	0.509784	0.570526	0.034416	0.177294	-0.006257	0.078665	0.026597	...	0.152954
number of bathrooms	-0.516909	-0.026485	0.509784	1.000000	0.753517	0.080806	0.502924	0.060104	0.183789	-0.128232	...	0.498127
living area	-0.648127	-0.021958	0.570526	0.753517	1.000000	0.174420	0.354743	0.105837	0.287728	-0.063358	...	0.309602
lot area	-0.100269	0.004392	0.034416	0.080806	0.174420	1.000000	-0.004138	0.026282	0.078308	-0.008548	...	0.051615
number of floors	-0.312305	-0.010335	0.177294	0.502924	0.354743	-0.004138	1.000000	0.016316	0.020153	-0.269928	...	0.481565
waterfront present	-0.112937	0.012006	-0.006257	0.060104	0.105837	0.026282	0.016316	1.000000	0.400206	0.018644	...	-0.024226
number of views	0.293004	0.004782	0.078665	0.183789	0.287728	0.078308	0.020153	0.400206	1.000000	0.052533	...	-0.055357
condition of the house	-0.045061	-0.027402	0.026597	-0.128232	-0.063358	-0.008548	-0.269928	0.018644	0.052533	1.000000	...	-0.381718
grade of the house	-0.673448	-0.033097	0.352945	0.663054	0.761835	0.110546	0.463082	0.079831	0.254532	-0.152530	...	0.440358
Area of the house(excluding basement)	-0.565116	-0.015994	0.473599	0.684391	0.875793	0.183553	0.525643	0.071865	0.162672	-0.167695	...	0.419369
Area of the basement	-0.290806	-0.015711	0.300332	0.287190	0.441491	0.019755	-0.242976	0.085441	0.293062	0.180609	...	-0.138843
Built Year	-0.068645	-0.005869	0.152954	0.498127	0.309602	0.051615	0.481565	-0.024226	-0.055357	-0.381718	...	1.000000
Renovation Year	-0.109155	-0.011636	0.016132	0.049669	0.059400	0.006848	0.006705	0.085865	0.102944	-0.062126	...	-0.233683
Postal Code	0.294709	0.018243	-0.044156	-0.105546	-0.080303	0.070131	-0.129788	0.038318	0.039268	0.045334	...	-0.062349
Lattitude	-0.479334	-0.023327	-0.013163	0.031156	0.054518	-0.090983	0.050731	-0.021795	-0.004555	-0.002998	...	-0.143153
Longitude	-0.070841	-0.018231	0.135712	0.223904	0.240208	0.221432	0.127550	-0.047791	-0.079706	-0.121189	...	0.414591
living_area_renov	-0.599900	-0.032495	0.389855	0.570530	0.757571	0.149744	0.285093	0.085743	0.281452	-0.099743	...	0.328625
lot_area_renov	-0.089604	-0.000050	0.029400	0.078627	0.180312	0.706812	-0.010120	0.032055	0.072300	-0.004748	...	0.072874
Number of schools nearby	-0.004821	-0.004071	0.003397	0.002180	0.002370	-0.012671	-0.007579	0.001563	0.008004	-0.006939	...	-0.001631
Distance from the airport	-0.004542	0.011457	-0.006157	0.009206	0.002511	0.003291	0.016567	0.001448	-0.001657	-0.002136	...	-0.003968
Price	-0.773114	-0.027919	0.308460	0.531735	0.712169	0.081992	0.262732	0.263687	0.395973	0.041376	...	0.050307

23 rows × 23 columns

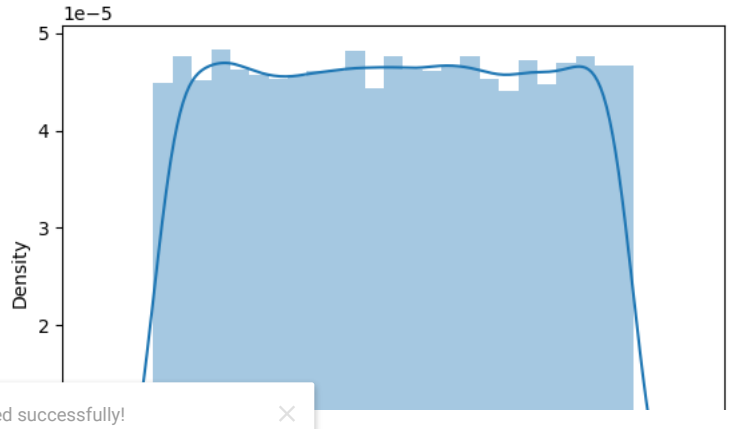
Saved successfully!

Univariant analysis

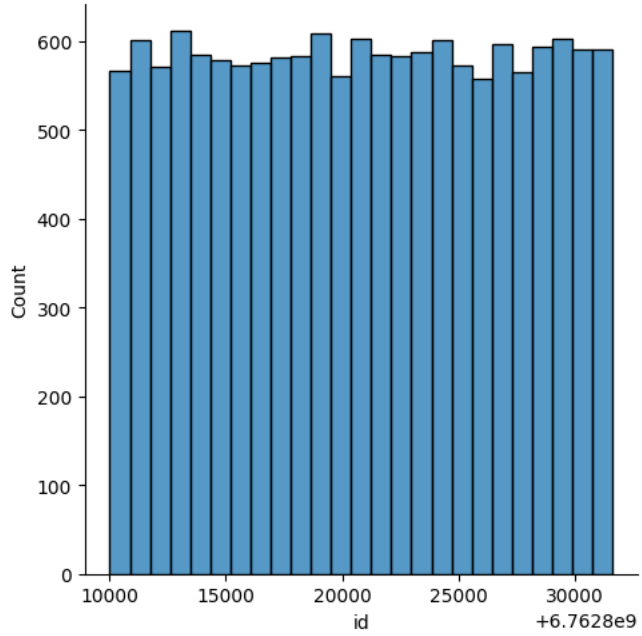
```
sns.distplot(df.id)
```

```
<ipython-input-25-ae12bbc1c629>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
  sns.distplot(df.id)
<Axes: xlabel='id', ylabel='Density'>
```

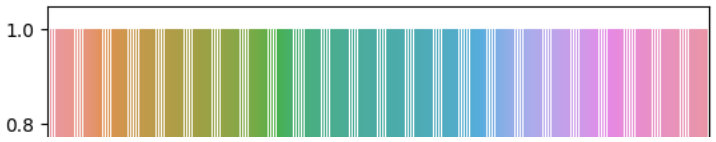



```
<seaborn.axisgrid.FacetGrid at 0x7fa3bc2ff0a0>
```



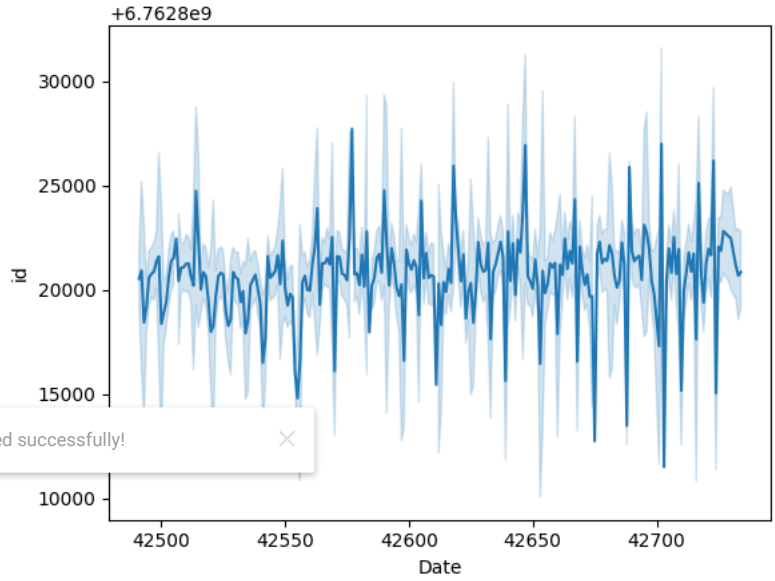

```
sns.barplot(x=df.id.value_counts().index,y=df.id.value_counts())
```

```
<Axes: ylabel='id'>
```




```
sns.lineplot(x=df.Date,y=df.id)
```

```
<Axes: xlabel='Date', ylabel='id'>
```

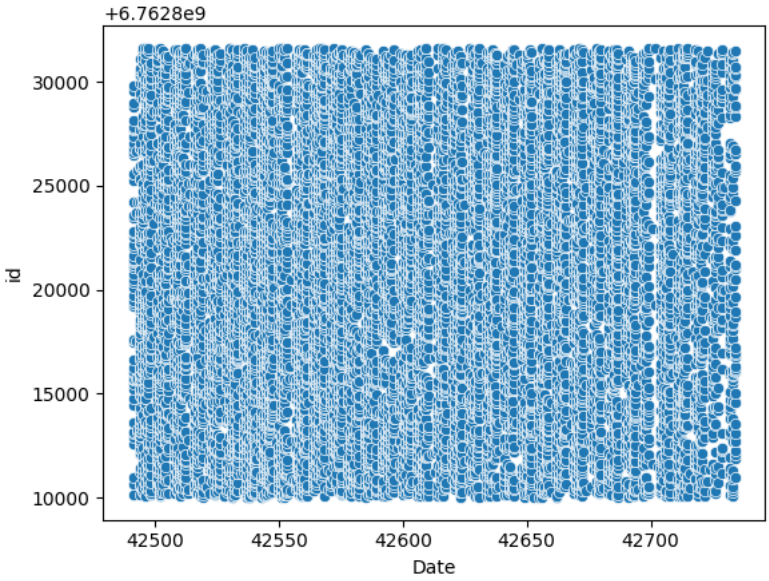


Saved successfully!

Bivariate Analysis

```
sns.scatterplot(x=df.Date,y=df.id)
```

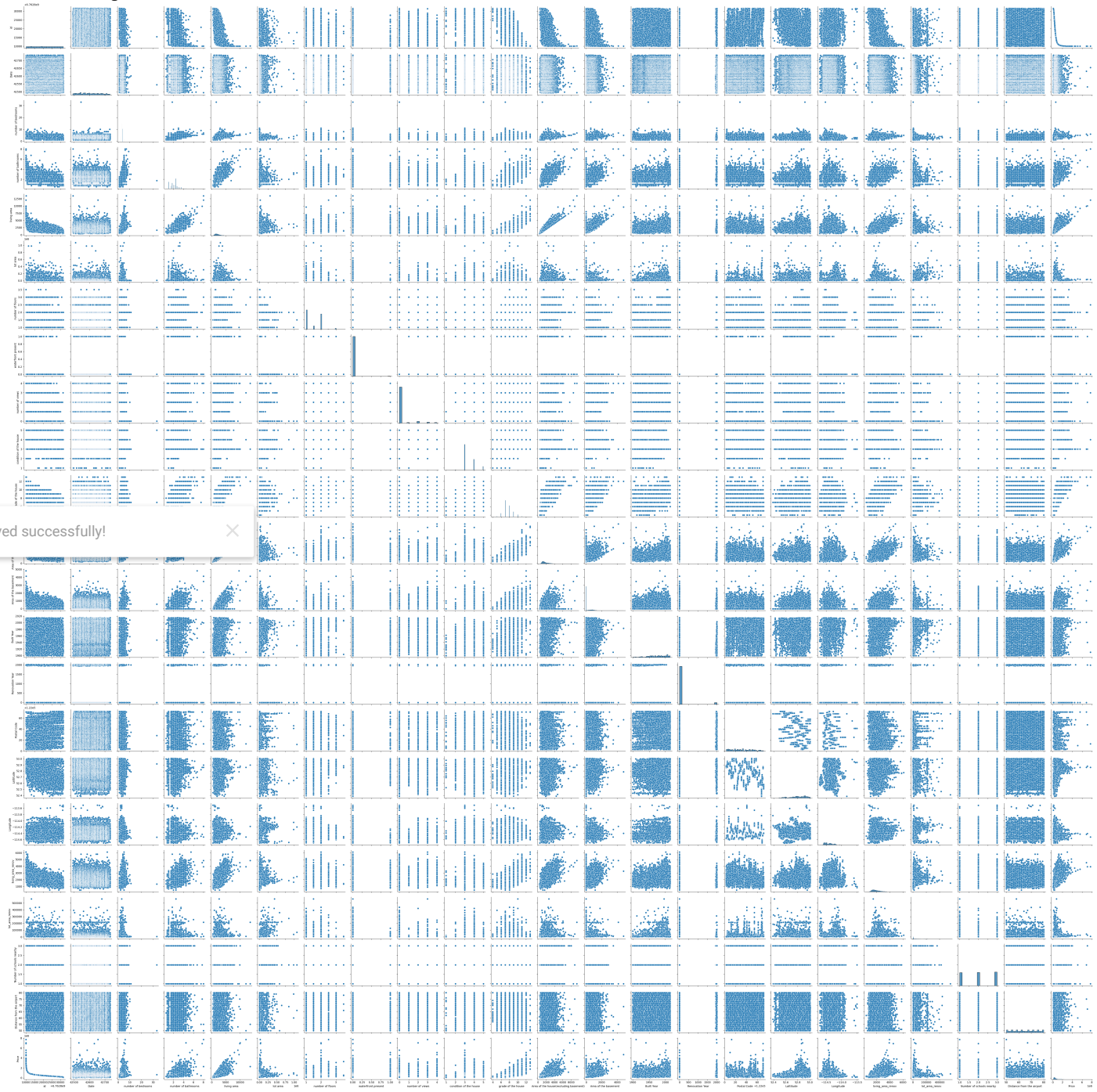
```
<Axes: xlabel='Date', ylabel='id'>
```



Multivariate Analysis

```
sns.pairplot(df)
```

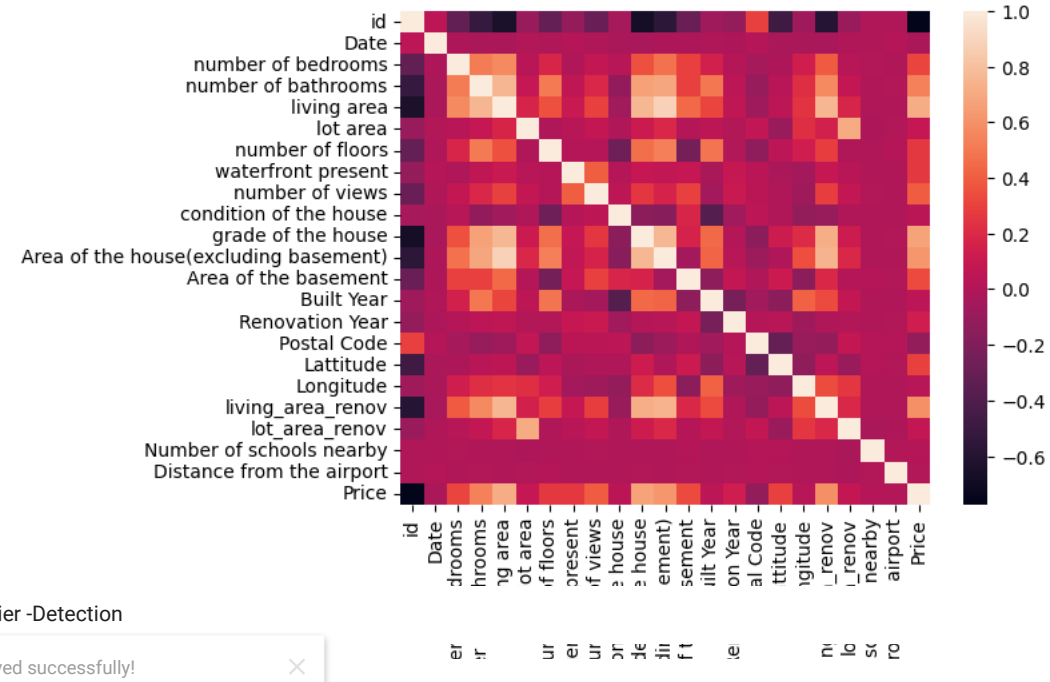
```
<seaborn.axisgrid.PairGrid at 0x7fa3b6a1a830>
```



Saved successfully!

```
sns.heatmap(df.corr())
```

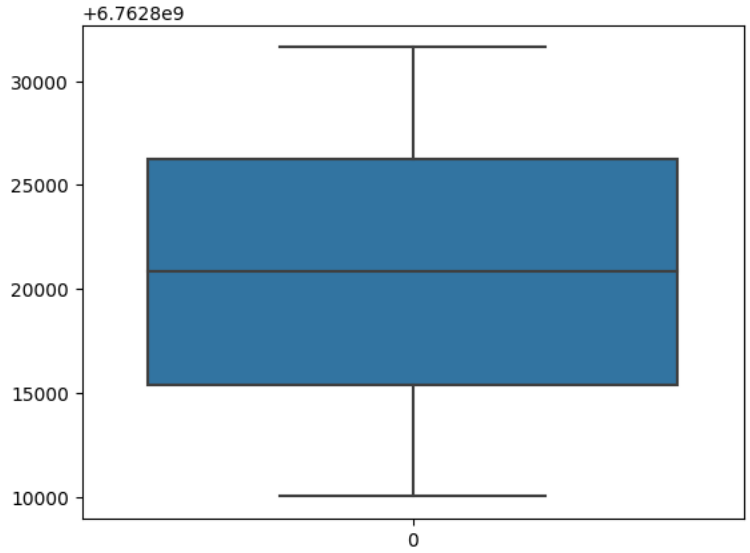
```
<Axes: >
```



Outlier -Detection

Saved successfully!

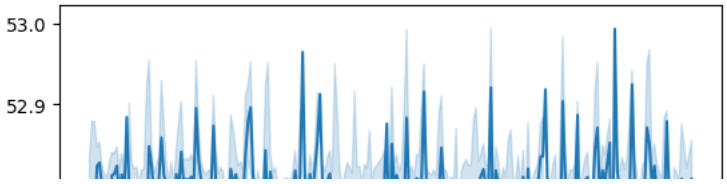
```
<Axes: >
```



Bivariate Analysis

```
sns.lineplot(x=df.Date,y=df.Lattitude)
```

```
<Axes: xlabel='Date', ylabel='Lattitude'>
```




```
## Outlier removal -IQR method,Z-score,Percentile
```

Outlier-Replacement-Median

```
q1 = df.id.quantile(0.25)  #(Q1)
q3 = df.id.quantile(0.75)  #(Q3)
```

```
print(q1)
print(q3)
```

```
6762815404.75
6762826251.25
```

```
IQR=q3-q1
```

Saved successfully!

```
lower_limit = q1-1.5 * IQR
```

```
Upper_limit
```

```
6762842521.0
```

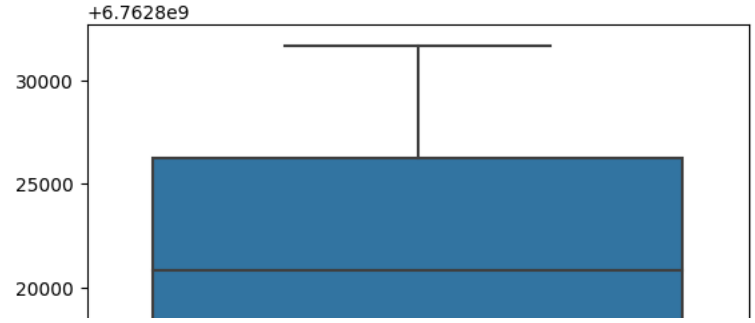
```
df.median()
```

```
id                                     6.762821e+09
Date                                   4.260000e+04
number of bedrooms                     3.000000e+00
number of bathrooms                    2.250000e+00
living area                            1.930000e+03
lot area                               7.620000e+03
number of floors                       1.500000e+00
waterfront present                     0.000000e+00
number of views                        0.000000e+00
condition of the house                 3.000000e+00
grade of the house                     7.000000e+00
Area of the house(excluding basement)  1.580000e+03
Area of the basement                   0.000000e+00
Built Year                             1.975000e+03
Renovation Year                        0.000000e+00
Postal Code                            1.220320e+05
Lattitude                              5.280640e+01
Longitude                             -1.144210e+02
living_area_renov                      1.850000e+03
lot_area_renov                         7.620000e+03
Number of schools nearby               2.000000e+00
Distance from the airport              6.500000e+01
Price                                  4.500000e+05
dtype: float64
```

```
df['id'] = np.where(df['id']>Upper_limit,30,df['id'])
```

```
sns.boxplot(df.id)
```

```
<Axes: >
```




```
df.shape
```

```
(14620, 23)
```

Encoding Technique

```
df.head()
```

Saved successfully!

	id	Date	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	Lattitud
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4	5	...	1921	0	122003	52.864
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0	5	...	1909	0	122004	52.887
2	6762810998	42491	5	2.75	2910	9480	1.5	0	0	3	...	1939	0	122004	52.885
3	6762812605	42491	4	2.50	3310	42998	2.0	0	0	3	...	2001	0	122005	52.953
4	6762812919	42491	3	2.00	2710	4500	1.5	0	0	4	...	1929	0	122006	52.904

5 rows × 23 columns

```
## label encoding

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

df.id = le.fit_transform(df.id)
df.Date= le.fit_transform(df.Date)

y=df.Date
y
```

```
0          0
1          0
2          0
3          0
4          0
         ...
14615    240
14616    240
14617    240
14618    240
14619    240
Name: Date, Length: 14620, dtype: int64
```

```
df.head()
```

	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	Long
0	84	0	5	2.50	3650	9050	2.0	0	4	5	...	1921	0	122003	52.8645	-11
1	398	0	4	2.50	2920	4000	1.5	0	0	5	...	1909	0	122004	52.8878	-11
2	647	0	5	2.75	2910	9480	1.5	0	0	3	...	1939	0	122004	52.8852	-11
3	1735	0	4	2.50	3310	42998	2.0	0	0	3	...	2001	0	122005	52.9532	-11
4	1951	0	3	2.00	2710	4500	1.5	0	0	4	...	1929	0	122006	52.9047	-11

5 rows × 23 columns

One hot Encoding Method

```
df1= pd.get_dummies(df,columns=['id'])
df1
```

Saved successfully!

	number	of	living area	lot area	number of floors	waterfront present	number of views	condition of the house	grade of the house	...	id_14610	id_14611	id_14612	id_1461
0	0	5	2.50	3650	9050	2.0	0	4	5	10	...	0	0	0	
1	0	4	2.50	2920	4000	1.5	0	0	5	8	...	0	0	0	
2	0	5	2.75	2910	9480	1.5	0	0	3	8	...	0	0	0	
3	0	4	2.50	3310	42998	2.0	0	0	3	9	...	0	0	0	
4	0	3	2.00	2710	4500	1.5	0	0	4	8	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
14615	240	2	1.50	1556	20000	1.0	0	0	4	7	...	0	0	0	
14616	240	3	2.00	1680	7000	1.5	0	0	4	7	...	0	0	0	
14617	240	2	1.00	1070	6120	1.0	0	0	3	6	...	0	0	0	
14618	240	4	1.00	1030	6621	1.0	0	0	4	6	...	0	0	0	
14619	240	3	1.00	900	4770	1.0	0	0	3	6	...	0	0	0	

14620 rows × 14642 columns

X and y split

```
X=df.drop(columns=['Price'],axis = 1)
X.head()
```

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Area of the basement	Built Year	Renovation Year	Postal Code	Latti
0	84	0	5	2.50	2650	9050	2.0	0	4	5		280	1921	0	122003	52

```
y = df.Price
y
```

```
0        2380000
1        1400000
2        1200000
3         838000
4         805000
          ...
14615     221700
14616     219200
14617     209000
14618     205000
14619     146000
Name: Price, Length: 14620, dtype: int64
```

Scaling

```
## MinMax Scaling
```

Saved successfully!

```
nMaxScaler
scale =MinMaxScaler()

X_scaled= pd.DataFrame(scale.fit_transform(X),columns = X.columns)

X_scaled.head()
```

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Area of the basement	Built Year	Renovation Year	P
0	0.005746	0.0	0.12500	0.266667	0.249051	0.007945	0.4	0.0	1.0	1.00	...	0.058091	0.182609	0.0	0.0
1	0.027225	0.0	0.09375	0.266667	0.193622	0.003241	0.2	0.0	0.0	1.00	...	0.209544	0.078261	0.0	0.0
2	0.044257	0.0	0.12500	0.300000	0.192863	0.008345	0.2	0.0	0.0	0.50	...	0.000000	0.339130	0.0	0.0
3	0.118681	0.0	0.09375	0.266667	0.223235	0.039562	0.4	0.0	0.0	0.50	...	0.000000	0.878261	0.0	0.0
4	0.133456	0.0	0.06250	0.200000	0.177677	0.003707	0.2	0.0	0.0	0.75	...	0.172199	0.252174	0.0	0.0

5 rows × 22 columns

Train and Split

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.3,random_state = 0)

x_train.shape
```

```
(10234, 22)
```

```
x_test.shape
```

```
(4386, 22)
```

```
y_test.shape
```

```
(4386,)
```

```
from keras.models import Sequential
from keras.layers import Dense
```

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
np.histogram([1, 2])
```

```
(array([1, 0, 0, 0, 0, 0, 0, 0, 0, 1]),
 array([1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2. ]))
```

```
x = df.drop('Price',axis=1)
x.head()
```

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Area of the basement	Built Year	Renovation Year	Postal Code	Latti
0	1	0	3	2.50	3650	9050	2.0	0	4	5	...	280	1921	0	122003	52.8
1	398	0	4	2.50	2920	4000	1.5	0	0	5	...	1010	1909	0	122004	52.8
2	647	0	5	2.75	2910	9480	1.5	0	0	3	...	0	1939	0	122004	52.8
3	1735	0	4	2.50	3310	42998	2.0	0	0	3	...	0	2001	0	122005	52.9
4	1951	0	3	2.00	2710	4500	1.5	0	0	4	...	830	1929	0	122006	52.9

5 rows × 22 columns

Saved successfully!

```
y = df['Price']
y
```

```
0        2380000
1        1400000
2        1200000
3         838000
4         805000
          ...
14615     221700
14616     219200
14617     209000
14618     205000
14619     146000
Name: Price, Length: 14620, dtype: int64
```

```
from sklearn.model_selection import train_test_split
```

```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,random_state=12)
```

Build an ANN Model

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 23 columns):
 #   Column                                Non-Null Count  Dtype
```

```
 ---  ------                                  --------------  -----
 0   id                                      14620 non-null  int64
 1   Date                                    14620 non-null  int64
 2   number of bedrooms                      14620 non-null  int64
 3   number of bathrooms                     14620 non-null  float64
 4   living area                             14620 non-null  int64
 5   lot area                                14620 non-null  int64
 6   number of floors                        14620 non-null  float64
 7   waterfront present                      14620 non-null  int64
 8   number of views                         14620 non-null  int64
 9   condition of the house                  14620 non-null  int64
 10  grade of the house                      14620 non-null  int64
 11  Area of the house(excluding basement)  14620 non-null  int64
 12  Area of the basement                    14620 non-null  int64
 13  Built Year                              14620 non-null  int64
 14  Renovation Year                         14620 non-null  int64
 15  Postal Code                             14620 non-null  int64
 16  Lattitude                               14620 non-null  float64
 17  Longitude                               14620 non-null  float64
 18  living_area_renov                       14620 non-null  int64
 19  lot_area_renov                          14620 non-null  int64
 20  Number of schools nearby                14620 non-null  int64
 21  Distance from the airport               14620 non-null  int64
 22  Price                                   14620 non-null  int64
dtypes: float64(4), int64(19)
memory usage: 2.6 MB
```

Saved successfully!

```
                                        0
Date                                    0
number of bedrooms                      0
number of bathrooms                     0
living area                             0
lot area                                0
number of floors                        0
waterfront present                      0
number of views                         0
condition of the house                  0
grade of the house                      0
Area of the house(excluding basement)   0
Area of the basement                    0
Built Year                              0
Renovation Year                         0
Postal Code                             0
Lattitude                               0
Longitude                               0
living_area_renov                       0
lot_area_renov                          0
Number of schools nearby                0
Distance from the airport               0
Price                                   0
dtype: int64
```

```
df['id'].unique()
```

```
array([   84,   398,   647, ..., 13943, 14015, 14519])
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
df['Date'].unique()
```

```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
```

```
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 23 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   id                                    14620 non-null  int64  
 1   Date                                  14620 non-null  int64  
 2   number of bedrooms                    14620 non-null  int64  
 3   number of bathrooms                   14620 non-null  float64
 4   living area                           14620 non-null  int64  
 5   lot area                              14620 non-null  int64  
 6   number of floors                      14620 non-null  float64
 7   waterfront present                    14620 non-null  int64  
 8   number of views                       14620 non-null  int64  
 9   condition of the house                14620 non-null  int64  
 10  grade of the house                    14620 non-null  int64  
                                           14620 non-null  int64  
                                           14620 non-null  int64  
                                           14620 non-null  int64  
 14  Renovation Year                       14620 non-null  int64  
 15  Postal Code                           14620 non-null  int64  
 16  Lattitude                             14620 non-null  float64
 17  Longitude                             14620 non-null  float64
 18  living_area_renov                     14620 non-null  int64  
 19  lot_area_renov                        14620 non-null  int64  
 20  Number of schools nearby              14620 non-null  int64  
 21  Distance from the airport             14620 non-null  int64  
 22  Price                                 14620 non-null  int64  
dtypes: float64(4), int64(19)
memory usage: 2.6 MB
```



```
x = df.drop('Price',axis=1)
x.head()
```

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Area of the basement	Built Year	Renovation Year	Postal Code	Latti
0	84	0	5	2.50	3650	9050	2.0	0	4	5	...	280	1921	0	122003	52.
1	398	0	4	2.50	2920	4000	1.5	0	0	5	...	1010	1909	0	122004	52.
2	647	0	5	2.75	2910	9480	1.5	0	0	3	...	0	1939	0	122004	52.
3	1735	0	4	2.50	3310	42998	2.0	0	0	3	...	0	2001	0	122005	52.
4	1951	0	3	2.00	2710	4500	1.5	0	0	4	...	830	1929	0	122006	52.

5 rows × 22 columns

```
y = df['Price']
y
```

```
0        2380000
1        1400000
2        1200000
3         838000
4         805000
          ...
14615     221700
14616     219200
14617     209000
14618     205000
```

```
14619     146000
Name: Price, Length: 14620, dtype: int64
```

```
from sklearn.model_selection import train_test_split

xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,random_state=12)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

reg = Sequential()
reg.add(Dense(4,activation='relu'))
reg.add(Dense(12,activation='relu'))
reg.add(Dense(32,activation='relu'))
reg.add(Dense(1,activation='linear'))

reg.compile(optimizer='adam',loss='mse',metrics=['mse'])
```



```
reg.fit(xtrain,ytrain,batch_size=10,epochs=25,validation_data=(xtest,ytest))
```

```
Epoch 1/25
1170/1170 [==============================] - 4s 3ms/step - loss: 148238516224.0000 - mse: 148238516224.0000 - val_loss: 154917945344.000
Epoch 2/25
1170/1170 [==============================] - 3s 2ms/step - loss: 66609639424.0000 - mse: 66609639424.0000 - val_loss: 67012669440.0000 -
Epoch 3/25
1170/1170 [==============================] - 3s 2ms/step - loss: 32393545728.0000 - mse: 32393545728.0000 - val_loss: 51977404416.0000 -
Epoch 4/25
1170/1170 [==============================] - 4s 3ms/step - loss: 24200749056.0000 - mse: 24200749056.0000 - val_loss: 37960912896.0000 -
Epoch 5/25
1170/1170 [==============================] - 3s 2ms/step - loss: 19964098560.0000 - mse: 19964100608.0000 - val_loss: 32543971328.0000 -
Epoch 6/25
1170/1170 [==============================] - 3s 2ms/step - loss: 17879166976.0000 - mse: 17879166976.0000 - val_loss: 28656603136.0000 -
Epoch 7/25
1170/1170 [==============================] - 3s 3ms/step - loss: 16022565888.0000 - mse: 16022565888.0000 - val_loss: 25155653632.0000 -
Epoch 8/25
1170/1170 [==============================] - 7s 6ms/step - loss: 14798630912.0000 - mse: 14798630912.0000 - val_loss: 23124426752.0000 -
Epoch 9/25
1170/1170 [==============================] - 4s 4ms/step - loss: 13942464512.0000 - mse: 13942464512.0000 - val_loss: 21679075328.0000 -
Epoch 10/25
1170/1170 [==============================] - 5s 4ms/step - loss: 13321062400.0000 - mse: 13321062400.0000 - val_loss: 20917897216.0000 -
Epoch 11/25
1170/1170 [==============================] - 7s 6ms/step - loss: 12731298816.0000 - mse: 12731298816.0000 - val_loss: 19960096768.0000 -
Epoch 12/25
1170/1170 [==============================] - 5s 4ms/step - loss: 12406204416.0000 - mse: 12406204416.0000 - val_loss: 18496010240.0000 -
Epoch 13/25
1170/1170 [==============================] - 3s 3ms/step - loss: 11933257728.0000 - mse: 11933257728.0000 - val_loss: 17950363648.0000 -
Epoch 14/25
1170/1170 [==============================] - 3s 3ms/step - loss: 11688812544.0000 - mse: 11688813568.0000 - val_loss: 17430949888.0000 -
Epoch 15/25
1170/1170 [==============================] - 3s 2ms/step - loss: 11484267520.0000 - mse: 11484267520.0000 - val_loss: 17036865536.0000 -
Epoch 16/25
1170/1170 [==============================] - 3s 2ms/step - loss: 11198681088.0000 - mse: 11198681088.0000 - val_loss: 16964756480.0000 -
Epoch 17/25
1170/1170 [==============================] - 2s 2ms/step - loss: 11129534464.0000 - mse: 11129534464.0000 - val_loss: 16533778432.0000 -
Epoch 18/25
1170/1170 [==============================] - 3s 2ms/step - loss: 10773383168.0000 - mse: 10773383168.0000 - val_loss: 16317324288.0000 -
Epoch 19/25
1170/1170 [==============================] - 3s 3ms/step - loss: 10875949056.0000 - mse: 10875949056.0000 - val_loss: 16965533696.0000 -
Epoch 20/25
1170/1170 [==============================] - 3s 3ms/step - loss: 10398380032.0000 - mse: 10398380032.0000 - val_loss: 16015109120.0000 -
Epoch 21/25
1170/1170 [==============================] - 2s 2ms/step - loss: 10635054080.0000 - mse: 10635054080.0000 - val_loss: 15396625408.0000 -
Epoch 22/25
1170/1170 [==============================] - 3s 2ms/step - loss: 10403016704.0000 - mse: 10403016704.0000 - val_loss: 16796754944.0000 -
Epoch 23/25
1170/1170 [==============================] - 2s 2ms/step - loss: 10356554752.0000 - mse: 10356554752.0000 - val_loss: 17184329728.0000 -
Epoch 24/25
1170/1170 [==============================] - 3s 3ms/step - loss: 10177088512.0000 - mse: 10177088512.0000 - val_loss: 16087503872.0000 -
Epoch 25/25
```

```
1170/1170 [==============================] - 3s 3ms/step - loss: 10232511488.0000 - mse: 10232511488.0000 - val_loss: 16210734080.0000 -
<keras.callbacks.History at 0x7fa30988a5f0>
```

```
reg.predict([[123123,345345,2343241,2]])
```

```
1/1 [==============================] - 1s 626ms/step
array([[806956.7]], dtype=float32)
```

```
# Handle missing values
df.fillna(0,inplace=True)
```

```
# Split the dataset into input and target variables
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
# Build the ANN model
model = Sequential()
                                    'relu', input_dim=X_train.shape[1]))
                                    'relu'))
model.add(Dense(units=1, activation='linear'))
```



```
# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
# Train the model
model.fit(X_train, y_train, batch_size=32, epochs=100)
```

```
366/366 [==============================] - 1s 2ms/step - loss: 10895004672.0000
Epoch 93/100
366/366 [==============================] - 1s 3ms/step - loss: 10796452864.0000
Epoch 94/100
366/366 [==============================] - 1s 3ms/step - loss: 10686840832.0000
Epoch 95/100
366/366 [==============================] - 1s 3ms/step - loss: 10557347840.0000
Epoch 96/100
366/366 [==============================] - 1s 4ms/step - loss: 10259372032.0000
Epoch 98/100
366/366 [==============================] - 1s 4ms/step - loss: 10175884288.0000
Epoch 99/100
366/366 [==============================] - 1s 3ms/step - loss: 10178080768.0000
Epoch 100/100
366/366 [==============================] - 1s 2ms/step - loss: 9940421632.0000
<keras.callbacks.History at 0x7fa309f51ed0>
```

Test the model

```
import tensorflow as tf


# Test the model
y_pred = model.predict(X_test)
mse = tf.keras.losses.mean_squared_error(y_test, y_pred).numpy()
print('Mean Squared Error:', mse)
```

Saved successfully!

```
======] - 0s 1ms/step
+11 9.4334589e+11 1.5283644e+11 ... 1.3892839e+11 2.0528737e+11
```

0s completed at 1:58 PM