```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot  as plt
import seaborn as sns
%matplotlib inline


import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


dataset = pd.read_csv('/content/data.csv')
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           4600 non-null   object 
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64  
 5   sqft_lot       4600 non-null   int64  
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64  
 8   view           4600 non-null   int64  
 9   condition      4600 non-null   int64  
 10  sqft_above     4600 non-null   int64  
 11  sqft_basement  4600 non-null   int64  
 12  yr_built       4600 non-null   int64  
 13  yr_renovated   4600 non-null   int64  
 14  street         4600 non-null   object 
 15  city           4600 non-null   object 
 16  statezip       4600 non-null   object 
 17  country        4600 non-null   object 
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

```
dataset.shape
```

```
(4600, 18)
```

```
dataset.head()
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfrc
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	

Delete date column Date column is irrelevant

```
dataset.drop(['date'], axis = 1, inplace = True)
dataset.head()
```

Checking how many different Countries are there

0	313000.0	3.0	1.50	1340	7912	1.5	0	0

```
dataset.country.value_counts()
```

```
USA    4600
Name: country, dtype: int64
```

```
dataset.drop(['country'], axis = 1, inplace = True)
dataset.head()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
0	313000.0	3.0	1.50	1340	7912	1.5	0	0
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4

```
dataset.drop(['street', 'city'], axis = 1, inplace = True)
dataset.head()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
0	313000.0	3.0	1.50	1340	7912	1.5	0	0
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4
2	342000.0	3.0	2.00	1930	11947	1.0	0	0

Checking for null values

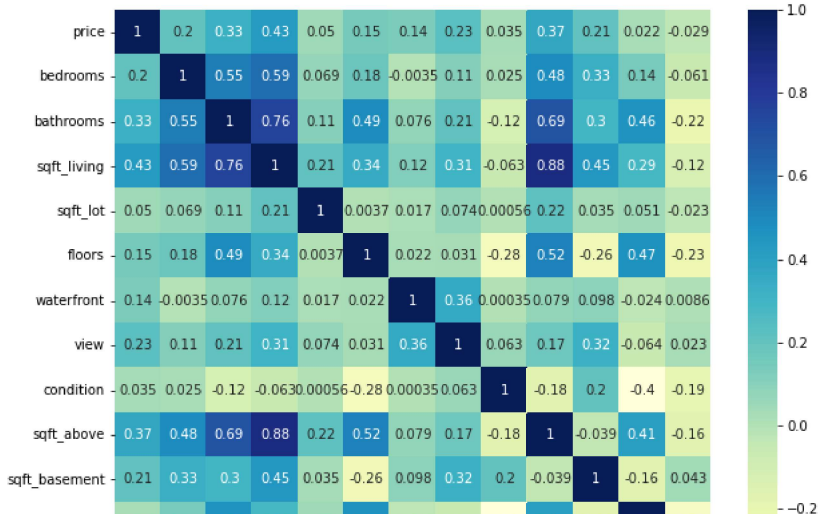
```
dataset.isnull().sum()
```

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

General corellation analysis

```
a4_dims = (10, 8)
fig, ax = plt.subplots(figsize=a4_dims)
cor = dataset.corr()
sns.heatmap(cor, annot = True, cmap="YlGnBu")
```

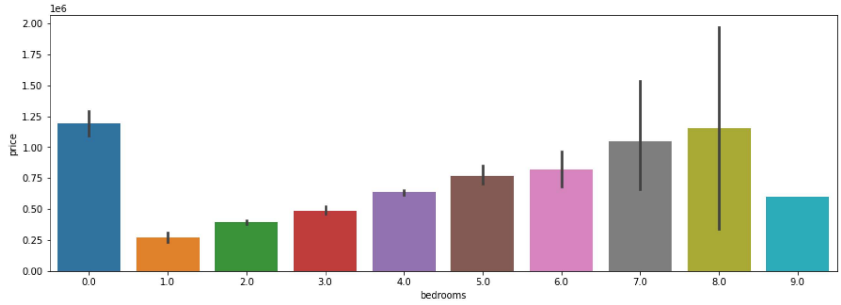
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe94ac14d50>
```



Analysis on number of bedroom feature

```
a4_dims = (15, 5)
fig, ax = plt.subplots(figsize=a4_dims)
sns.barplot(x = dataset.bedrooms, y = dataset.price)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe947bd98d0>
```




```
dataset.groupby('bedrooms').price.agg([len, min, max])
```

bedrooms	len	min	max
0.0	2	1095000.0	1295648.0
1.0	38	0.0	540000.0
2.0	566	0.0	1695000.0
3.0	2032	0.0	26590000.0
4.0	1531	0.0	4489000.0
5.0	353	0.0	7062500.0
6.0	61	0.0	3100000.0
7.0	14	280000.0	3200000.0
8.0	2	340000.0	1970000.0
9.0	1	599999.0	599999.0

```
df = dataset[(dataset.bedrooms > 0) & (dataset.bedrooms < 9)].copy()
df.shape
```

```
(4597, 14)
```

Analysis on the zipcode feature

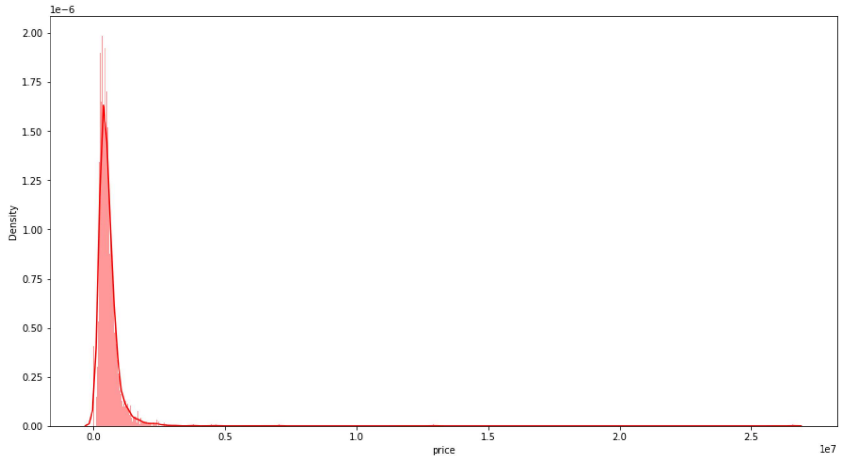
```
df.statezip.value_counts()
```

```
WA 98103    147
WA 98052    135
WA 98117    132
WA 98115    130
WA 98006    110
            ...
WA 98047      6
WA 98288      3
WA 98050      2
WA 98354      2
WA 98068      1
Name: statezip, Length: 77, dtype: int64
```

```
a4_dims = (5, 18)
fig, ax = plt.subplots(figsize=a4_dims)
sns.barplot(ax = ax, x = df.price, y = df.statezip)
```

```
a4_dims = (15, 8)
fig, ax = plt.subplots(figsize=a4_dims)
sns.distplot(a = df.price, bins = 1000, color = 'r', ax = ax)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarnin
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fe946309fd0>
```




```
df.price.agg([min, max])
```

```
min           0.0
max    26590000.0
Name: price, dtype: float64
```

How many instances are there with price = 0?

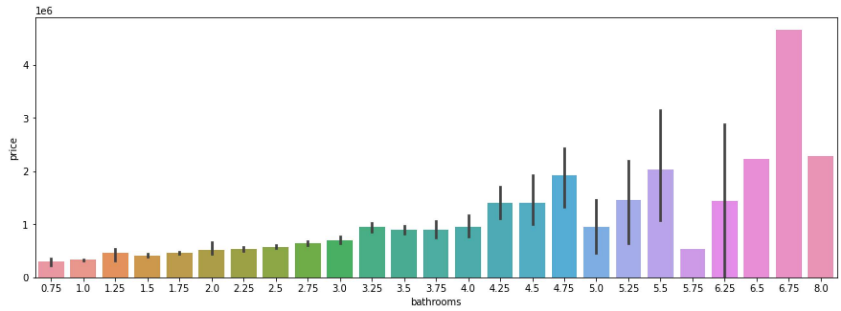
```
len(df[(df.price == 0)])
```

```
49
```

Analysis on bathroom feature w.r.t. price

```
a4_dims = (15, 5)
fig, ax = plt.subplots(figsize=a4_dims)
sns.barplot(x = df.bathrooms, y = df.price)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe9454ab3d0>
```



Analysis on all the instances whose price is 0

```
zero_price = df[(df.price == 0)].copy()
zero_price.shape
```

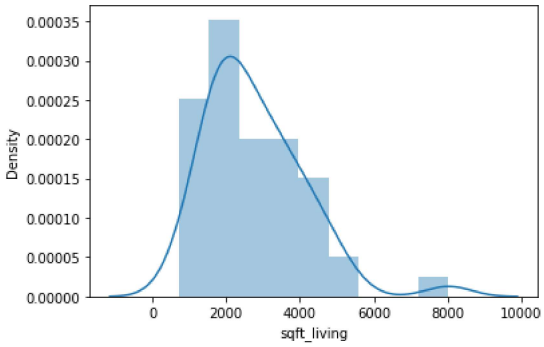
```
(49, 14)
```

```
zero_price.head()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
4354	0.0	3.0	1.75	1490	10125	1.0	0	0
4356	0.0	4.0	2.75	2600	5390	1.0	0	0
4357	0.0	6.0	2.75	3200	9200	1.0	0	2

```
sns.distplot(zero_price.sqft_living)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarnin
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fe945ff6850>
```




```
zero_price.agg([min, max, 'mean', 'median'])
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
min	0.0	1.000000	1.00000	720.000000	3500.000000	1.0	0.000000
max	0.0	6.000000	6.25000	8020.000000	188200.000000	3.0	1.000000
mean	0.0	3.979592	2.69898	2787.142857	16453.306122	1.5	0.061224

```
sim_from_ori = df[(df.bedrooms == 4) & (df.bathrooms > 1) & (df.bathrooms < 4) & (df.sqft_living > 2500) & (df.sqft_living < 3000) & (df.floo
sim_from_ori.shape
```

```
(79, 14)
```

```
sim_from_ori.head()
```

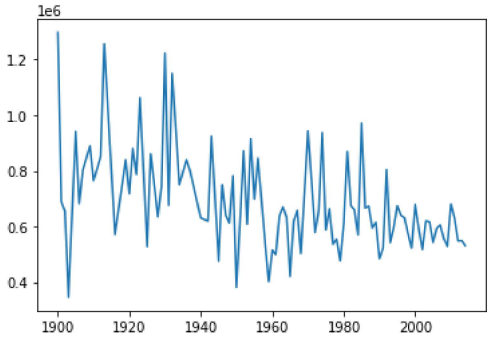
	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	vie
11	1400000.0	4.0	2.50	2920	4000	1.5	0	
172	407000.0	4.0	2.25	2810	23400	1.0	0	
207	360000.0	4.0	2.00	2680	18768	1.0	0	

```
sim_from_ori.price.mean()
```

```
735475.0370705189
```

```
yr_sqft = df[(df.sqft_living > 2499) & (df.sqft_living < 2900)].copy()
yr_price_avg = yr_sqft.groupby('yr_built').price.agg('mean')
plt.plot(yr_price_avg)
```

```
[<matplotlib.lines.Line2D at 0x7fe945175bd0>]
```




```
df.price.replace(to_replace = 0, value = 735000, inplace = True)
len(df[(df.price == 0)])
```

```
0
```

```
df.head()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
0	313000.0	3.0	1.50	1340	7912	1.5	0	0
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4
2	342000.0	3.0	2.00	1930	11947	1.0	0	0

Feature reduction

```
df.drop(['sqft_above'], axis = 1, inplace = True)
df.shape
```

```
(4597, 13)
```

Handling the index order

```
df = df.reset_index()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4597 entries, 0 to 4596
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   index             4597 non-null   int64  
 1   price             4597 non-null   float64
 2   bedrooms          4597 non-null   float64
 3   bathrooms         4597 non-null   float64
 4   sqft_living       4597 non-null   int64  
 5   sqft_lot          4597 non-null   int64  
 6   floors            4597 non-null   float64
 7   waterfront        4597 non-null   int64  
 8   view              4597 non-null   int64  
 9   condition         4597 non-null   int64  
 10  sqft_basement     4597 non-null   int64  
 11  yr_built          4597 non-null   int64  
 12  yr_renovated      4597 non-null   int64  
 13  statezip          4597 non-null   object 
 14  statezip_encoded  4597 non-null   int64
```

```
dtypes: float64(4), int64(10), object(1)
memory usage: 538.8+ KB
```

Handling categorical statezip feature

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df['statezip_encoded'] = le.fit_transform(df.statezip)
df.head()
```

	index	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
0	0	313000.0	3.0	1.50	1340	7912	1.5	0
1	1	2384000.0	5.0	2.50	3650	9050	2.0	0
2	2	342000.0	3.0	2.00	1930	11947	1.0	0

```
df.statezip_encoded.value_counts()
```

```
47    147
31    135
56    132
54    130
5     110
     ...
28      6
75      3
29      2
76      2
39      1
Name: statezip_encoded, Length: 77, dtype: int6447    147
31    135
56    132
54    130
5     110
     ...
28      6
75      3
29      2
76      2
39      1
Name: statezip_encoded, Length: 77, dtype: int64
```

```
df.drop(['statezip'], axis = 1, inplace = True)
df.head()
```

	index	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
0	0	313000.0	3.0	1.50	1340	7912	1.5	0
1	1	2384000.0	5.0	2.50	3650	9050	2.0	0
2	2	342000.0	3.0	2.00	1930	11947	1.0	0
3	3	420000.0	3.0	2.25	2000	8030	1.0	0
4	4	550000.0	4.0	2.50	1940	10500	1.0	0

5 rows × 91 columns

```
from sklearn.preprocessing import OneHotEncoder
ohc = OneHotEncoder()
ohc_df = pd.DataFrame(ohc.fit_transform(df[['statezip_encoded']]).toarray())
# ohc_df = ohc_df.astype(int)
ohc_df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	67	68	69	70	71	72	73
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 77 columns

	0	1	2	3	4	5	6	7	8	9	...	67	68	69	70	71	72	73
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
df = df.join(ohc_df)
df.head()
```

	index	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
0	0	313000.0	3.0	1.50	1340	7912	1.5	0
1	1	2384000.0	5.0	2.50	3650	9050	2.0	0
2	2	342000.0	3.0	2.00	1930	11947	1.0	0
3	3	420000.0	3.0	2.25	2000	8030	1.0	0
4	4	550000.0	4.0	2.50	1940	10500	1.0	0

5 rows × 92 columns

```
df.tail()
```

	index	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wat
4592	4595	308166.666667	3.0	1.75	1510	6360	1.0	
4593	4596	534333.333333	3.0	2.50	1460	7573	2.0	
4594	4597	416904.166667	3.0	2.50	3010	7014	2.0	
4595	4598	203400.000000	4.0	2.00	2090	6630	1.0	
4596	4599	220600.000000	3.0	2.50	1490	8102	2.0	

5 rows × 91 columns

```
df.drop(['statezip_encoded'], axis = 1, inplace = True)
df.info
```

```
<bound method DataFrame.info of       index         price  bedrooms  bathrooms  sqft_living  sqft_lot  floors  \
0         0  3.130000e+05       3.0       1.50         1340      7912     1.5
1         1  2.384000e+06       5.0       2.50         3650      9050     2.0
2         2  3.420000e+05       3.0       2.00         1930     11947     1.0
3         3  4.200000e+05       3.0       2.25         2000      8030     1.0
4         4  5.500000e+05       4.0       2.50         1940     10500     1.0
...     ...           ...       ...        ...          ...       ...     ...
4592   4595  3.081667e+05       3.0       1.75         1510      6360     1.0
4593   4596  5.343333e+05       3.0       2.50         1460      7573     2.0
4594   4597  4.169042e+05       3.0       2.50         3010      7014     2.0
4595   4598  2.034000e+05       4.0       2.00         2090      6630     1.0
4596   4599  2.206000e+05       3.0       2.50         1490      8102     2.0

      waterfront  view  condition  ...   67   68   69   70   71   72   73  \
0              0     0          3  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1              0     4          5  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2              0     0          4  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3              0     0          4  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4              0     0          4  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
...          ...   ...        ...  ...  ...  ...  ...  ...  ...  ...  ...
4592           0     0          4  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4593           0     0          3  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4594           0     0          3  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4595           0     0          3  ...  0.0  0.0  0.0  0.0  1.0  0.0  0.0
4596           0     0          4  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0

       74   75   76
```

```
0     0.0  0.0  0.0
1     0.0  0.0  0.0
2     0.0  0.0  0.0
3     0.0  0.0  0.0
4     0.0  0.0  0.0
...   ...  ...  ...
4592  0.0  0.0  0.0
4593  0.0  0.0  0.0
4594  0.0  0.0  0.0
4595  0.0  0.0  0.0
4596  0.0  0.0  0.0

[4597 rows x 90 columns]>
```

Splitting into train and test set

```
df.shape
```

```
(4597, 90)
```

```
X = df.iloc[:, 1:]
X.shape
```

```
(4597, 89)
```

```
y = df.price
```

```
from sklearn.model_selection import train_test_split
X_train, X_rem, y_train, y_rem = train_test_split(X, y, test_size=0.1, random_state=42)
print(len(X_train) / len(df))
```

```
0.8999347400478573
```

```
X_val, X_test, y_val, y_test = train_test_split(X_rem, y_rem, test_size=0.5, random_state=42)
print(len(X_test) / len(y_rem))
```

```
0.5
```

```
print(len(X_train))
print(len(X_val))
print(len(X_val))
```

```
4137
230
230
```

Linear regression

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()

lin_reg.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all st
  FutureWarning,
LinearRegression()
```

```
from sklearn.metrics import mean_squared_error
y_pred = lin_reg.predict(X_val)
mse = mean_squared_error(y_pred, y_val)
rmse = np.sqrt(mse)
rmse
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all st
  FutureWarning,
1.2578489139925118e-09
```

```
y_val.head(10)
```

```
1073    175000.0
4524    950100.0
4434    309487.5
2572    427000.0
4310    375000.0
4017    665000.0
4241    759000.0
3139    425000.0
2283    325000.0
4200    679000.0
Name: price, dtype: float64
```

```
y_pred
```

```
array([  175000.        ,   950100.        ,   309487.5       ,
         427000.        ,   375000.        ,   665000.        ,
         759000.        ,   425000.        ,   325000.        ,
         679000.        ,   336000.        ,   630000.        ,
         492000.        ,   842500.        ,   264000.        ,
         383962.        ,   425000.        ,   925000.        ,
         235000.        ,   648360.        ,   580000.        ,
         615000.        ,   277000.        ,   225000.        ,
        2000000.        ,   565000.        ,   740000.        ,
         455000.        ,   385500.        ,   564000.        ,
         645000.        ,   583000.        ,   360000.        ,
         280000.        ,   285000.        ,   638000.        ,
         223000.        ,   239950.        ,   450000.        ,
         400000.        ,   375000.        ,   970000.        ,
         560000.        ,   825000.        ,   888550.        ,
         440000.        ,   199950.        ,   705380.        ,
         285000.        ,   220000.        ,   480500.        ,
         453246.        ,   716500.        ,   282766.666667  ,
         735000.        ,   547500.        ,   245000.        ,
         523950.        ,   318989.        ,   762400.        ,
         589950.        ,   225000.        ,   339000.        ,
         479000.        ,   475000.        ,   326500.        ,
        1532500.        ,   365000.        ,   450800.        ,
         452500.        ,   608000.        ,   420000.        ,
         405500.        ,   290000.        ,   710000.        ,
         625000.        ,   665000.        ,   257500.        ,
         371000.        ,   505000.        ,   647500.        ,
        1054690.        ,   527550.        ,   380000.        ,
         315000.        ,   210000.        ,   168000.        ,
         530000.        ,   525000.        ,   527000.        ,
         379950.        ,  1450000.        ,   201500.        ,
         285000.        ,  1225000.        ,   329950.        ,
         489950.        ,   815000.        ,   359000.        ,
         819900.        ,   980000.        ,   723000.        ,
         735000.        ,   290256.        ,   515000.        ,
         499000.        ,   455000.        ,   580000.        ,
         641000.        ,   860000.        ,   772000.        ,
         875000.        ,   530000.        ,   735000.        ,
         250000.        ,   740000.        ,   588500.        ,
        1320000.        ,   370000.        ,   197500.        ,
         330000.        ,   615000.        ,   283200.        ,
         612500.        ,  1160000.        ,   499500.        ,
         625000.        ,   200000.        ,   485000.        ,
         525000.        ,  1150000.        ,   484998.        ,
         687500.        ,   490000.        ,   210000.        ,
         280000.        ,   542500.        ,   294700.        ,
         355000.        ,   152000.        ,   910000.        ,
         415000.        ,   712000.        ,   875000.        ,
         540000.        ,   735000.        ,   724800.        ,
         590300.        ,   288400.        ,   570000.        ,
         490000.        ,   530000.        ,   375000.        ,
         669000.        ,  1300000.        ,   890000.        ,
         300000.        ,   671000.        ,   812000.        ,
         487028.        ,   285000.        ,   375000.        ,
         899000.        ,   405000.        ,   712000.        ,
         359950.        ,   690500.        ,   735000.        ,
        1325000.        ,  1100000.        ,   910000.        ,
         209950.        ,   550000.        ,   451000.        ,
```

```
y_pred_test = lin_reg.predict(X_test)
mse = mean_squared_error(y_pred_test, y_test)
rmse = np.sqrt(mse)
rmse
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all st
  FutureWarning,
```

```
3.1878327723836875e-10
```

```
lin_reg.score(X_test, y_test)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all st
  FutureWarning,
1.0
```

```
y_test
```

```
3454    450000.0
3857    300000.0
1818    174500.0
856     215000.0
1001    430000.0
          ...
3760    392000.0
2596    650000.0
1839    327500.0
1835    320000.0
787     317000.0
Name: price, Length: 230, dtype: float64
```

```
y_pred_test
```

```
array([ 450000.      ,  300000.      ,  174500.      ,  215000.      ,
        430000.      ,  421000.      ,  280000.      ,  380000.      ,
        260000.      ,  453500.      ,  299950.      ,  326983.333333,
        440000.      ,  270000.      ,  869000.      ,  681716.      ,
        100000.      ,  331366.666667,  540000.      ,  280000.      ,
        399500.      ,  330000.      ,  399950.      ,  250600.      ,
        640000.      ,  530000.      ,  395000.      ,  615000.      ,
        840000.      ,  805000.      ,  381000.      ,  625000.      ,
        624800.      ,  870000.      ,  315000.      ,  375000.      ,
        810000.      ,  660000.      , 1080000.      ,  274950.      ,
        440000.      ,  510000.      ,  386380.      ,  510000.      ,
        282508.888889,  860000.      ,  443000.      ,  459500.      ,
        545000.      ,  623000.      ,  211000.      ,  590000.      ,
        735000.      ,  275000.      ,  747500.      ,  630000.      ,
        345000.      ,  300000.      , 1228000.      ,  825500.      ,
        333000.      ,  540000.      ,  320000.      ,  276900.      ,
        751000.      ,  345000.      ,  459950.      ,  540000.      ,
        310000.      ,  465000.      ,  840000.      ,  535000.      ,
        763101.      ,  337000.      , 1405000.      ,  715000.      ,
        535000.      ,  395000.      , 2351956.      ,  369000.      ,
        235000.      ,  370000.      ,  592500.      ,  350000.      ,
       1150000.      ,  672500.      ,  500000.      ,  367500.      ,
        500000.      ,  475000.      ,  471000.      ,  335000.      ,
        785000.      ,  395000.      ,  459000.      ,  260000.      ,
        675000.      ,  320000.      ,  300000.      ,   83300.      ,
        550000.      ,  310000.      ,  450000.      ,  247875.      ,
        750000.      ,  419190.      ,  430000.      ,  285000.      ,
        425000.      ,  176225.      ,  579000.      ,  245000.      ,
        582000.      ,  775000.      ,  215000.      ,  543200.      ,
       1058000.      ,  259950.      ,  341000.      ,  330000.      ,
        585000.      ,  259000.      ,  534333.333333,  315000.      ,
        429000.      ,  947500.      ,  330000.      ,  515000.      ,
        285000.      ,  430000.      ,  235000.      ,  949880.      ,
       1925000.      ,  160000.      ,  505000.      ,  315000.      ,
        240000.      ,  645325.      ,  471000.      ,  397500.      ,
        329445.      ,  285000.      , 1600000.      ,  499000.      ,
        515500.      ,  418000.      ,  235000.      ,  250000.      ,
        350000.      ,  460000.      ,  413500.      ,  220600.      ,
        230000.      ,  396500.      ,  309000.      ,  442000.      ,
        325000.      , 1150000.      ,  790000.      ,  399950.      ,
        365000.      ,  351999.      ,  432000.      ,  895000.      ,
        360000.      ,  239950.      ,  435000.      ,  269950.      ,
        530000.      ,  737500.      ,  314950.      ,  655000.      ,
        936000.      , 1010000.      ,  551000.      ,  246500.      ,
        439000.      ,  700000.      ,  605000.      ,  402500.      ,
        470000.      , 1800000.      ,  400000.      ,  851000.      ,
        538888.      ,  770000.      , 1030000.      ,  875000.      ,
        460000.      ,  736500.      ,  589500.      ,  366750.      ,
        299000.      , 3800000.      ,  385000.      ,  675000.      ,
        193000.      ,  850000.      ,  386000.      ,  860000.      ,
        615000.      ,  305000.      ,  258000.      ,  230000.      ,
        350000.      ,  210000.      ,  749995.      ,  380000.      ,
       1105000.      ,  320000.      ,  480000.      ,  486445.833333,
        375000.      ,  400000.      ,  549900.      ,  312000.      ,
```

```
            650880.      ,  385000.      ,  735000.      ,  392500.      ,
            682000.      ,  367500.      ,  343000.      ,  433000.      ,
            577000.      ,  392000.      ,  650000.      ,  327500.      ,
            220000.      ,  217000.      ])
```

```
from sklearn.tree import DecisionTreeRegressor

reg = DecisionTreeRegressor(random_state = 42, max_depth = 10)
reg.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all st
  FutureWarning,
DecisionTreeRegressor(max_depth=10, random_state=42)
```

```
reg.score(X_test, y_test)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692: FutureWarning: Feature names only support names that are all st
  FutureWarning,
0.9997242025740952
```

```
y_val.head(10)
```

```
1073    175000.0
4524    950100.0
4434    309487.5
2572    427000.0
4310    375000.0
4017    665000.0
4241    759000.0
3139    425000.0
2283    325000.0
4200    679000.0
Name: price, dtype: float64
```