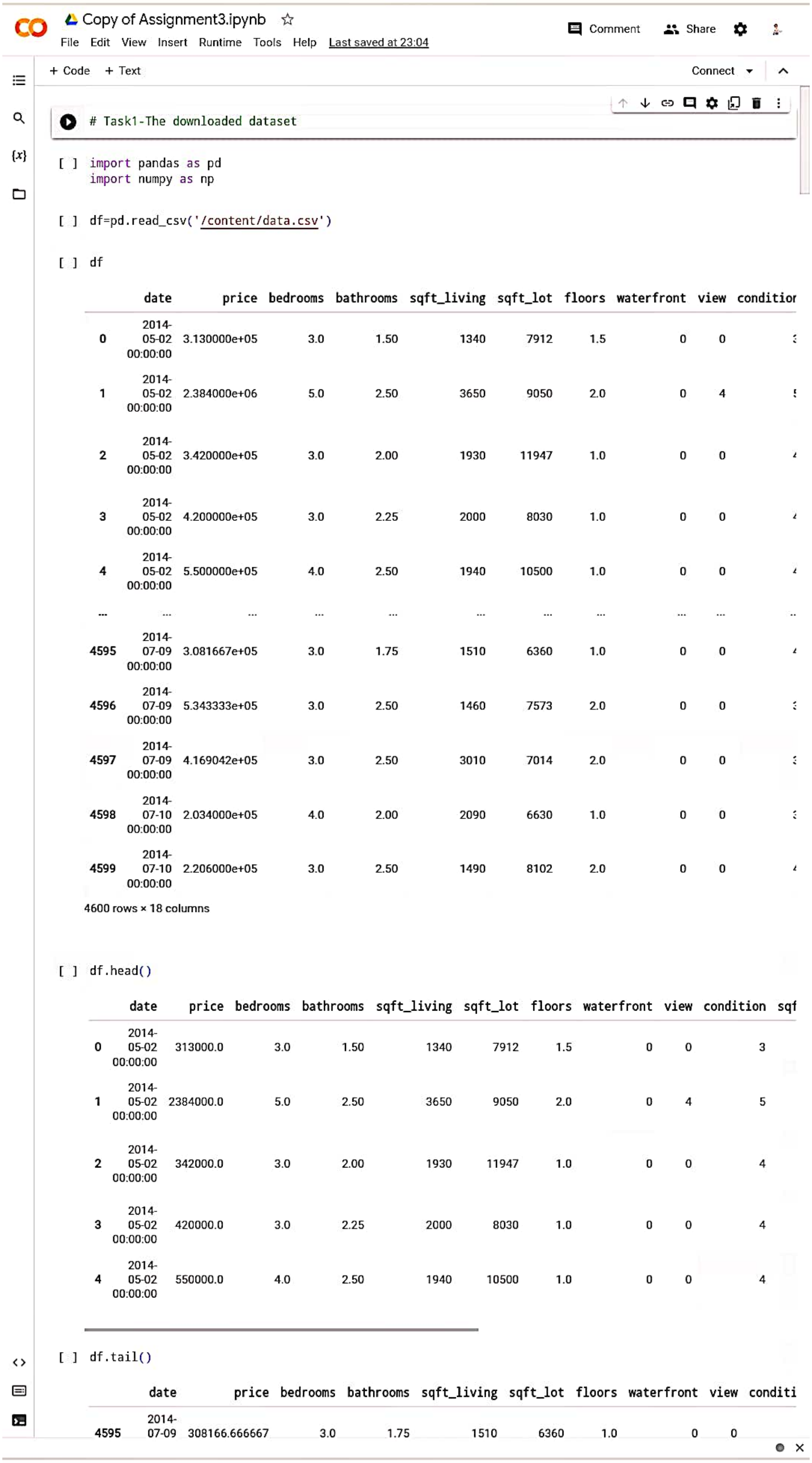
4G 4G 23:08 ♫ Vo) 1 LTE2 6.30 KB/s 10

b.research.google.com + 1 ⋮


CO
Copy of Assignment3.ipynb ☆
File Edit View Insert Runtime Tools Help Last saved at 23:04
Comment
Share
+ Code + Text
Connect

```
# Task1-The downloaded dataset
```

```
[ ] import pandas as pd
    import numpy as np
```

```
[ ] df=pd.read_csv('/content/data.csv')
```

```
[ ] df
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	conditior
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4
...	...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	4

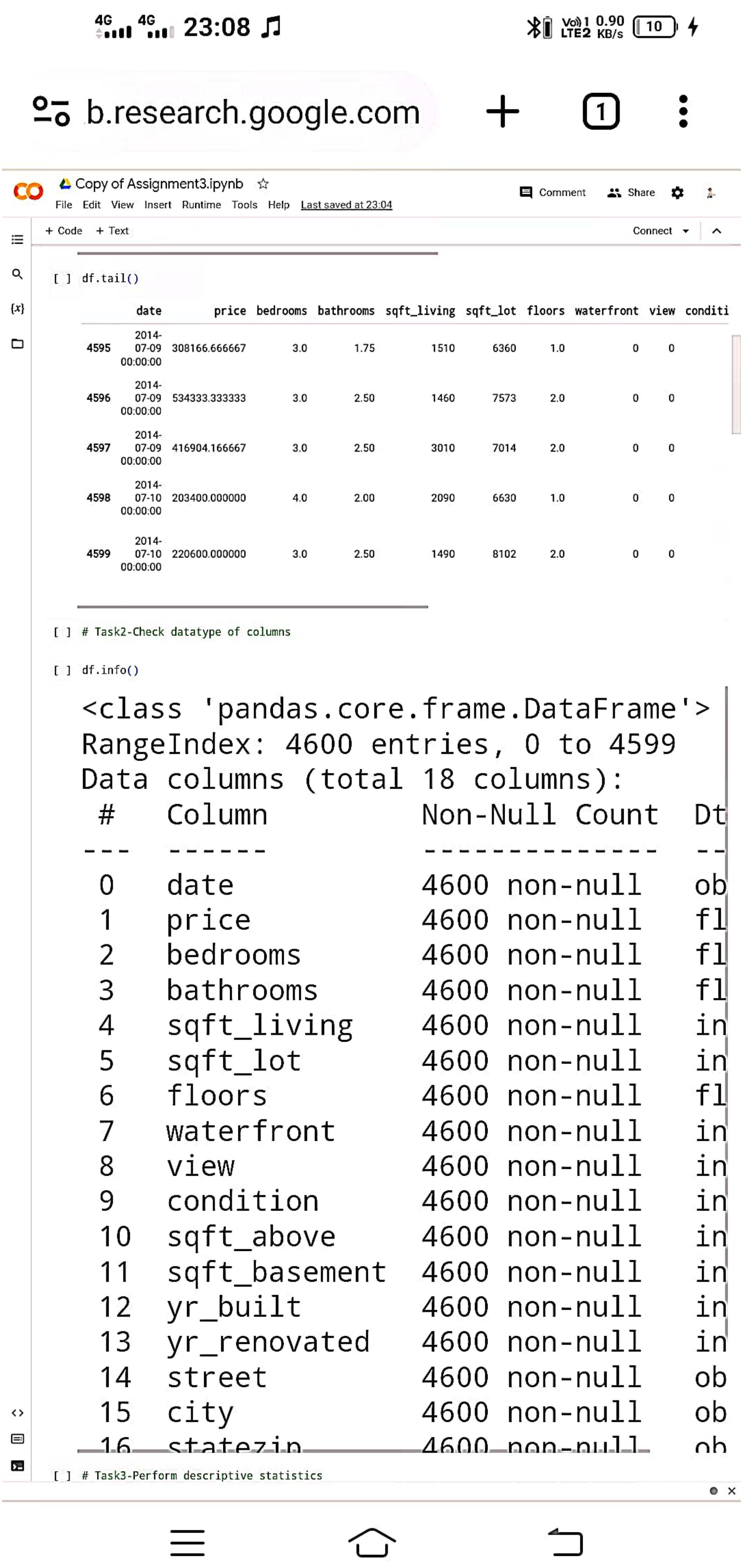
4600 rows × 18 columns

```
[ ] df.head()
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqf
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	

```
[ ] df.tail()
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	conditi
4595	2014-07-09	308166.666667	3.0	1.75	1510	6360	1.0	0	0	


23:08
b.research.google.com
Copy of Assignment3.ipynb
File Edit View Insert Runtime Tools Help Last saved at 23:04
Comment
Share
+ Code
+ Text
Connect

```
[ ] df.tail()
```

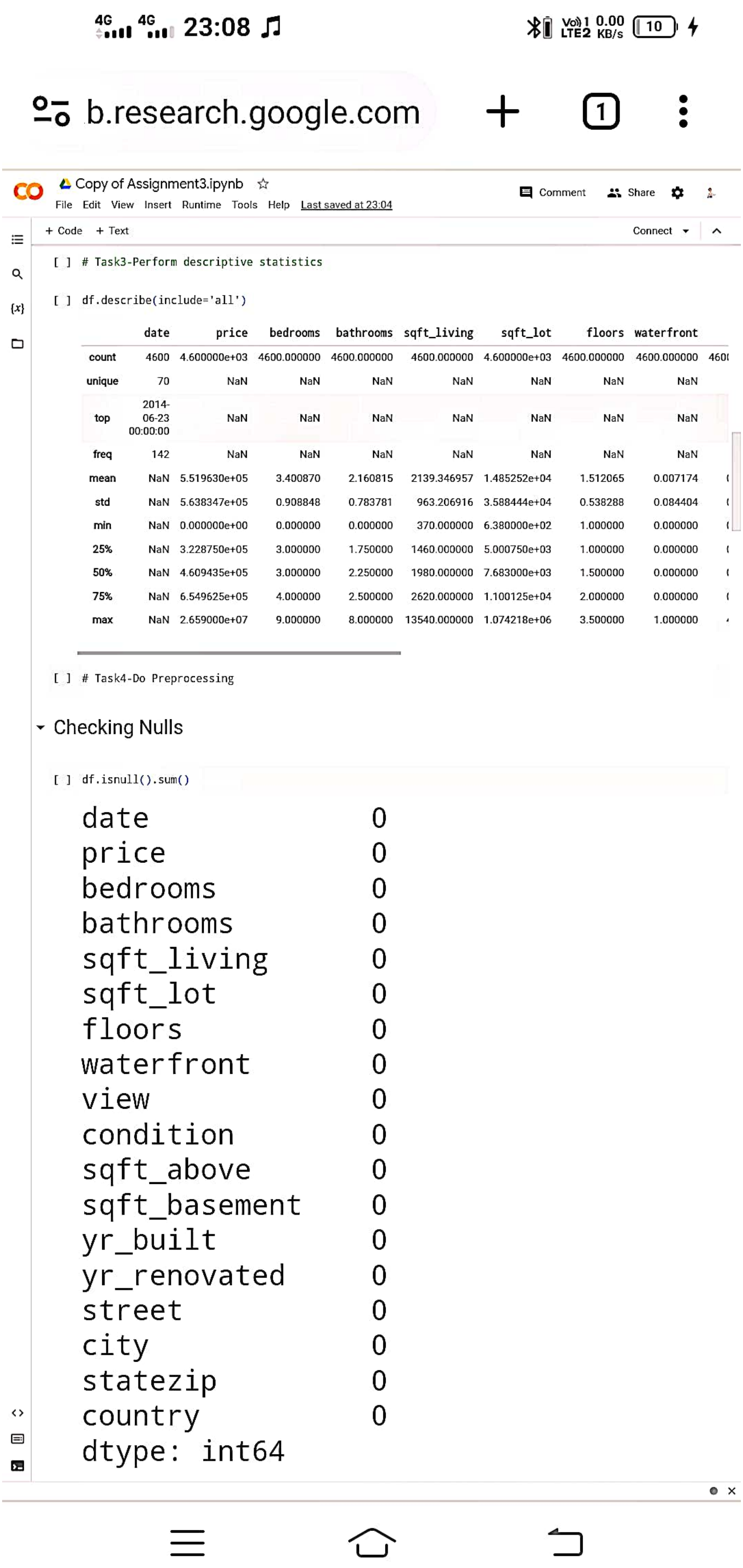
	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	conditi
4595	2014-07-09 00:00:00	308166.666667	3.0	1.75	1510	6360	1.0	0	0	
4596	2014-07-09 00:00:00	534333.333333	3.0	2.50	1460	7573	2.0	0	0	
4597	2014-07-09 00:00:00	416904.166667	3.0	2.50	3010	7014	2.0	0	0	
4598	2014-07-10 00:00:00	203400.000000	4.0	2.00	2090	6630	1.0	0	0	
4599	2014-07-10 00:00:00	220600.000000	3.0	2.50	1490	8102	2.0	0	0	

```
[ ] # Task2-Check datatype of columns
```

```
[ ] df.info()
```

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

```
[ ] # Task3-Perform descriptive statistics
```


23:08
b.research.google.com
Copy of Assignment3.ipynb
File Edit View Insert Runtime Tools Help Last saved at 23:04
Comment
Share
+ Code + Text
Connect

```
[ ] # Task3-Perform descriptive statistics

[ ] df.describe(include='all')
```

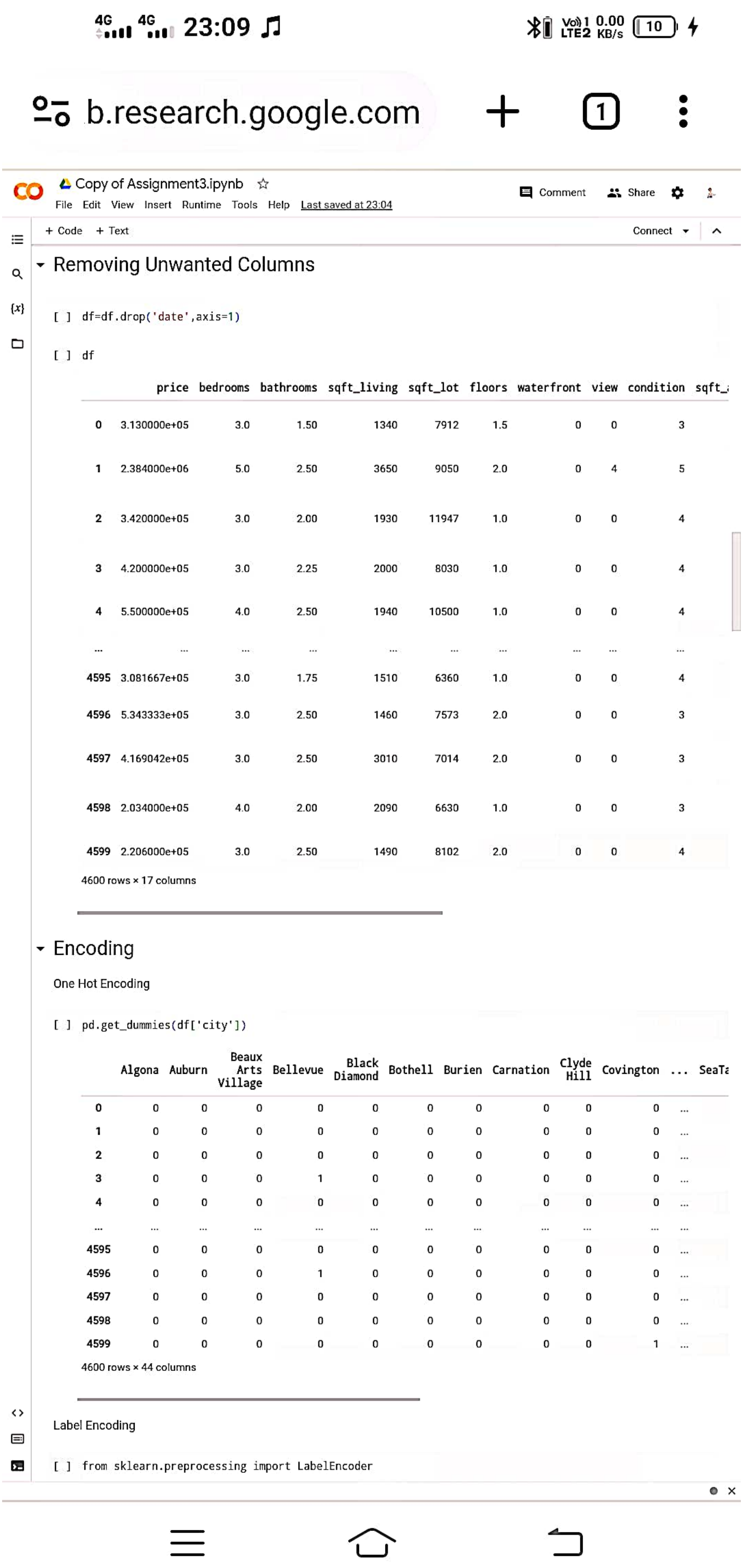
	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	
count	4600	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000	4600.000000	460(
unique	70	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
top	2014-06-23 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
freq	142	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
mean	NaN	5.519630e+05	3.400870	2.160815	2139.346957	1.485252e+04	1.512065	0.007174	(
std	NaN	5.638347e+05	0.908848	0.783781	963.206916	3.588444e+04	0.538288	0.084404	(
min	NaN	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000	(
25%	NaN	3.228750e+05	3.000000	1.750000	1460.000000	5.000750e+03	1.000000	0.000000	(
50%	NaN	4.609435e+05	3.000000	2.250000	1980.000000	7.683000e+03	1.500000	0.000000	(
75%	NaN	6.549625e+05	4.000000	2.500000	2620.000000	1.100125e+04	2.000000	0.000000	(
max	NaN	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000	4

```
[ ] # Task4-Do Preprocessing
```

Checking Nulls

```
[ ] df.isnull().sum()
```

```
date             0
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
street           0
city             0
statezip         0
country          0
dtype: int64
```



Removing Unwanted Columns

```
[ ] df=df.drop('date',axis=1)
```

```
[ ] df
```

```
             price  bedrooms  bathrooms  sqft_living  sqft_lot  floors  waterfront  view  condition  sqft_
0     3.130000e+05       3.0       1.50         1340      7912     1.5           0     0          3
1     2.384000e+06       5.0       2.50         3650      9050     2.0           0     4          5
2     3.420000e+05       3.0       2.00         1930     11947     1.0           0     0          4
3     4.200000e+05       3.0       2.25         2000      8030     1.0           0     0          4
4     5.500000e+05       4.0       2.50         1940     10500     1.0           0     0          4
...            ...       ...        ...          ...       ...     ...         ...   ...        ...
4595  3.081667e+05       3.0       1.75         1510      6360     1.0           0     0          4
4596  5.343333e+05       3.0       2.50         1460      7573     2.0           0     0          3
4597  4.169042e+05       3.0       2.50         3010      7014     2.0           0     0          3
4598  2.034000e+05       4.0       2.00         2090      6630     1.0           0     0          3
4599  2.206000e+05       3.0       2.50         1490      8102     2.0           0     0          4

4600 rows × 17 columns
```

Encoding

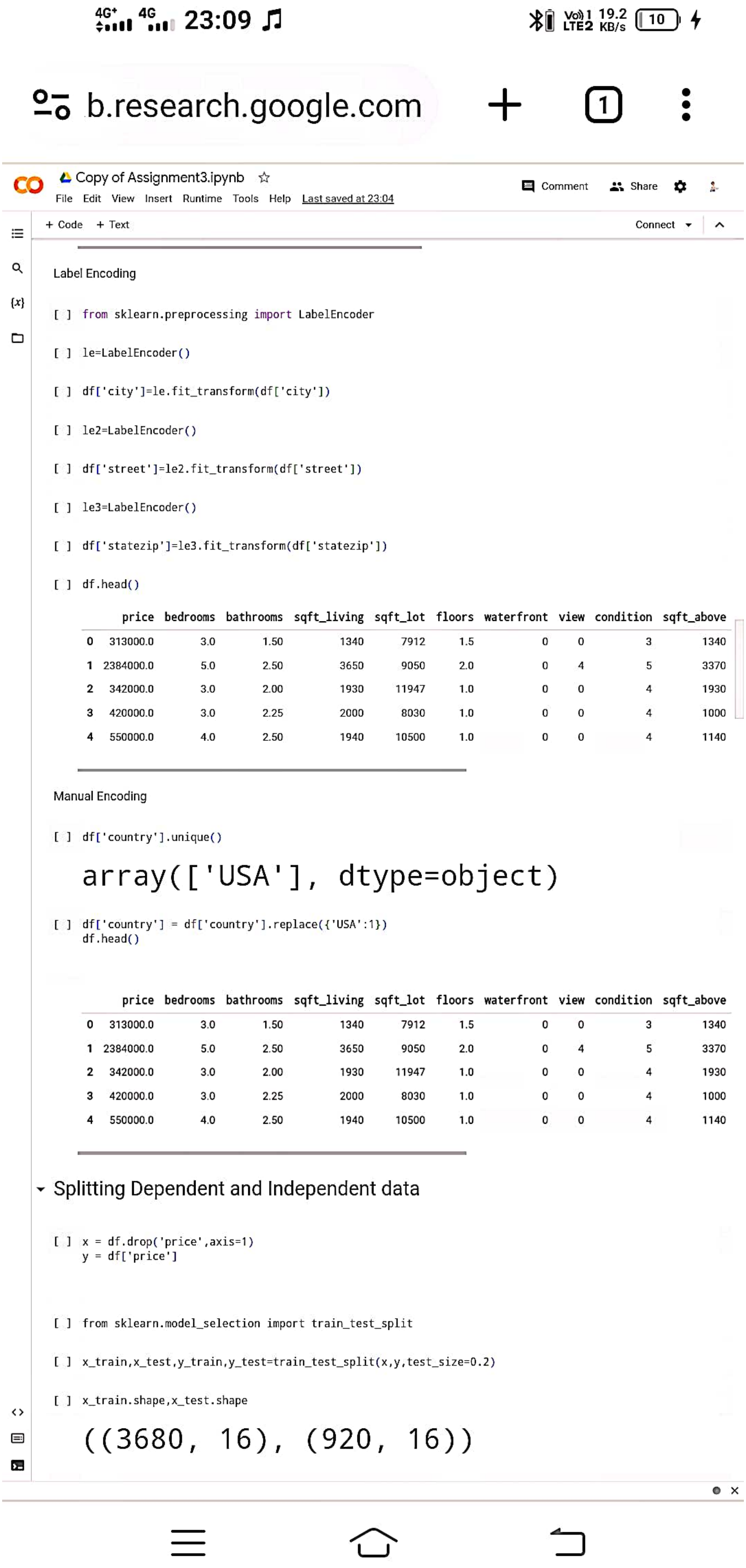
One Hot Encoding

```
[ ] pd.get_dummies(df['city'])
```

```
                     Beaux              Black                                  Clyde
      Algona  Auburn  Arts   Bellevue  Diamond  Bothell  Burien  Carnation  Hill   Covington  ...  SeaTa
                      Village
0          0       0      0         0        0        0       0          0      0          0  ...
1          0       0      0         0        0        0       0          0      0          0  ...
2          0       0      0         0        0        0       0          0      0          0  ...
3          0       0      0         1        0        0       0          0      0          0  ...
4          0       0      0         0        0        0       0          0      0          0  ...
...      ...     ...    ...       ...      ...      ...     ...        ...    ...        ...  ...
4595       0       0      0         0        0        0       0          0      0          0  ...
4596       0       0      0         1        0        0       0          0      0          0  ...
4597       0       0      0         0        0        0       0          0      0          0  ...
4598       0       0      0         0        0        0       0          0      0          0  ...
4599       0       0      0         0        0        0       0          0      0          1  ...

4600 rows × 44 columns
```

Label Encoding

```
[ ] from sklearn.preprocessing import LabelEncoder
```



Label Encoding

```
[ ] from sklearn.preprocessing import LabelEncoder
```

```
[ ] le=LabelEncoder()
```

```
[ ] df['city']=le.fit_transform(df['city'])
```

```
[ ] le2=LabelEncoder()
```

```
[ ] df['street']=le2.fit_transform(df['street'])
```

```
[ ] le3=LabelEncoder()
```

```
[ ] df['statezip']=le3.fit_transform(df['statezip'])
```

```
[ ] df.head()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140

Manual Encoding

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

```
array(['USA'], dtype=object)
```

```
[ ] df['country'] = df['country'].replace({'USA':1})
    df.head()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140

Splitting Dependent and Independent data

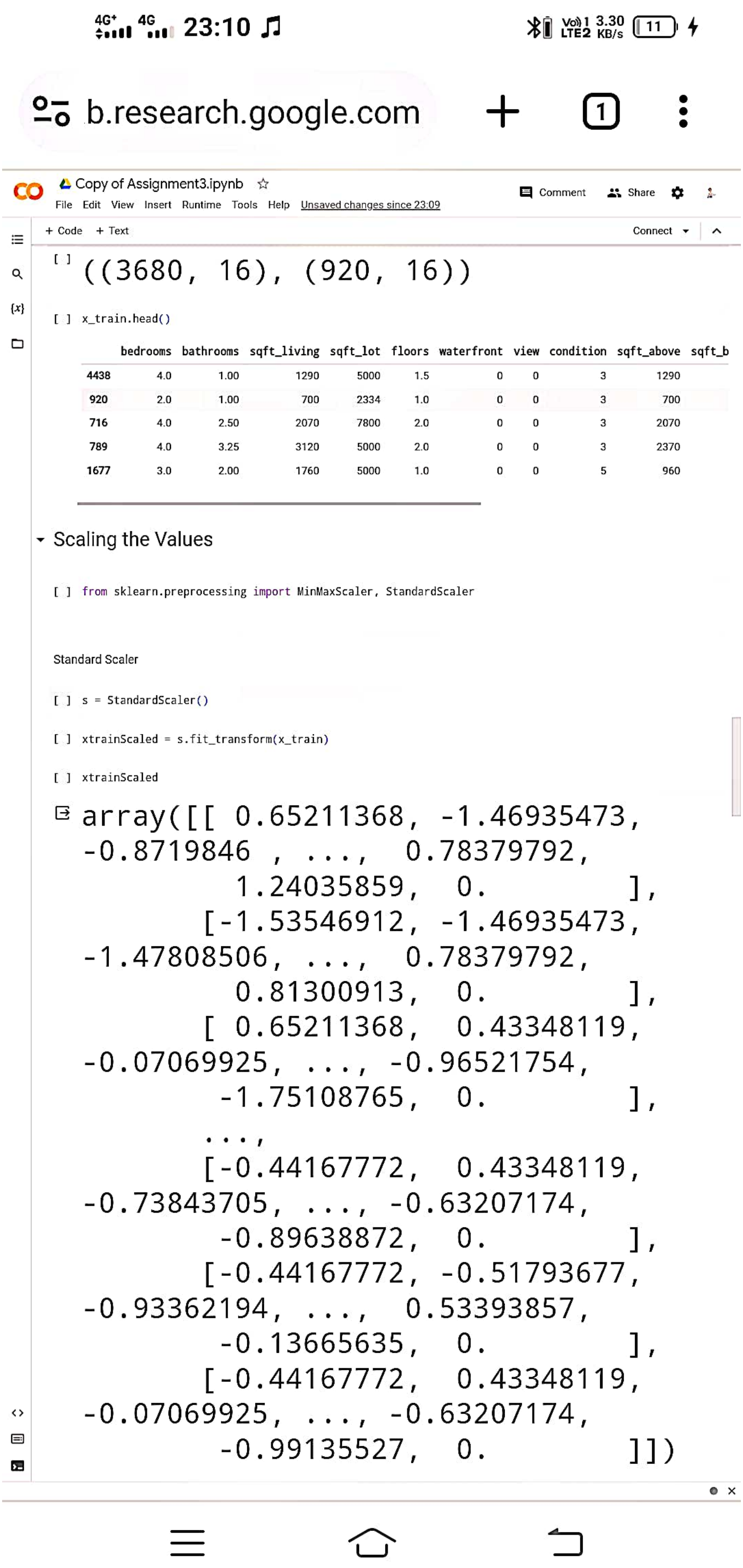
```
[ ] x = df.drop('price',axis=1)
    y = df['price']
```

```
[ ] from sklearn.model_selection import train_test_split
```

```
[ ] x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
[ ] x_train.shape,x_test.shape
```

```
((3680, 16), (920, 16))
```


4G+ 4G 23:10
3.30 KB/s 11
b.research.google.com
1
CO
Copy of Assignment3.ipynb
File Edit View Insert Runtime Tools Help Unsaved changes since 23:09
Comment
Share
+ Code + Text
Connect

```
[ ] ((3680, 16), (920, 16))
```

```
[ ] x_train.head()
```

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_b
4438	4.0	1.00	1290	5000	1.5	0	0	3	1290	
920	2.0	1.00	700	2334	1.0	0	0	3	700	
716	4.0	2.50	2070	7800	2.0	0	0	3	2070	
789	4.0	3.25	3120	5000	2.0	0	0	3	2370	
1677	3.0	2.00	1760	5000	1.0	0	0	5	960	

Scaling the Values

```
[ ] from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

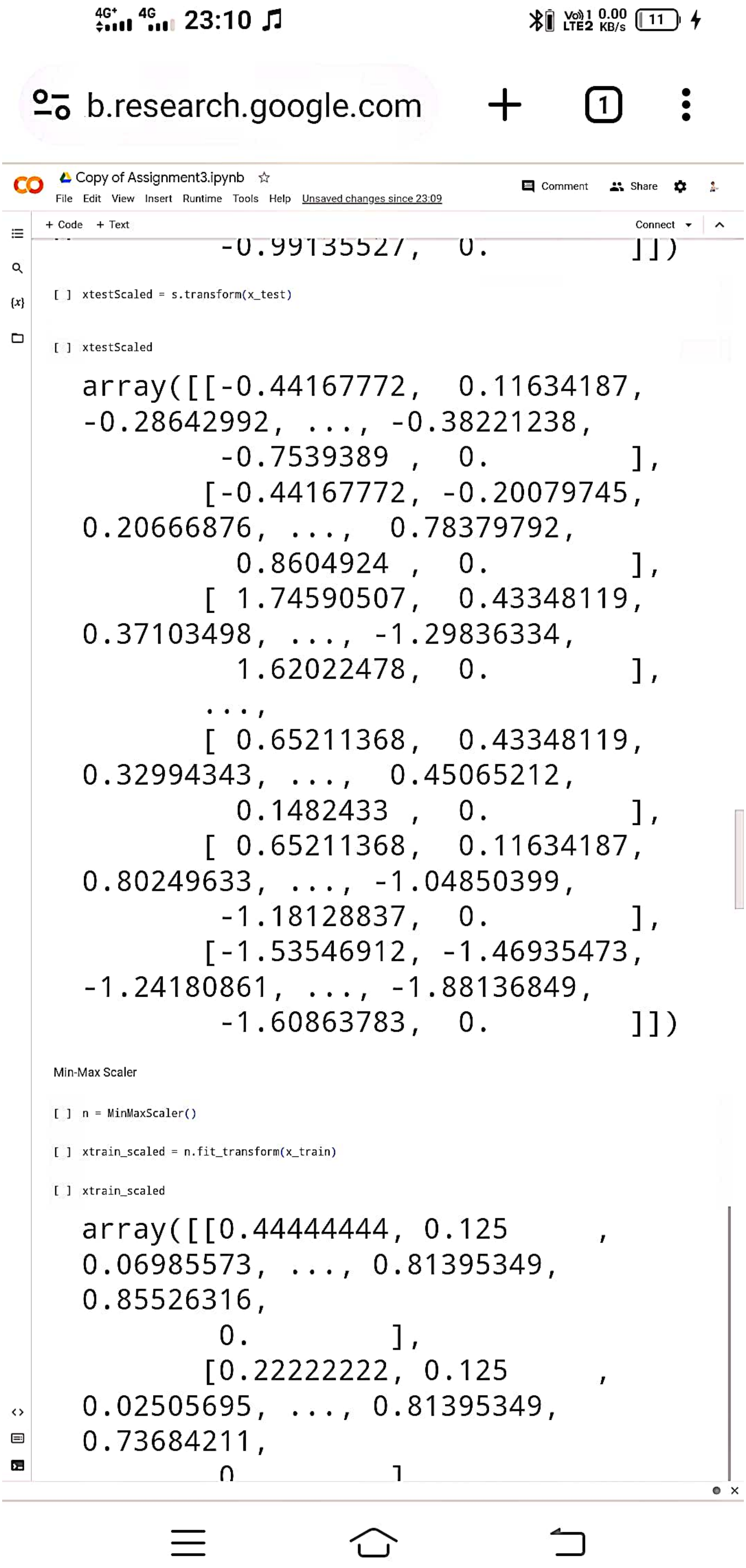
Standard Scaler

```
[ ] s = StandardScaler()
```

```
[ ] xtrainScaled = s.fit_transform(x_train)
```

```
[ ] xtrainScaled
```

```
array([[ 0.65211368, -1.46935473,
-0.8719846 , ...,  0.78379792,
          1.24035859,  0.        ],
        [-1.53546912, -1.46935473,
-1.47808506, ...,  0.78379792,
          0.81300913,  0.        ],
        [ 0.65211368,  0.43348119,
-0.07069925, ..., -0.96521754,
         -1.75108765,  0.        ],
        ...,
        [-0.44167772,  0.43348119,
-0.73843705, ..., -0.63207174,
         -0.89638872,  0.        ],
        [-0.44167772, -0.51793677,
-0.93362194, ...,  0.53393857,
         -0.13665635,  0.        ],
        [-0.44167772,  0.43348119,
-0.07069925, ..., -0.63207174,
         -0.99135527,  0.        ]])
```


23:10
b.research.google.com
Copy of Assignment3.ipynb
File Edit View Insert Runtime Tools Help Unsaved changes since 23:09
Comment
Share
+ Code + Text
Connect

```
        -0.99135527,   0.          ]])
```

```
[ ] xtestScaled = s.transform(x_test)
```

```
[ ] xtestScaled
```

```
array([[-0.44167772,  0.11634187,
-0.28642992, ..., -0.38221238,
        -0.7539389 ,  0.          ],
       [-0.44167772, -0.20079745,
0.20666876, ...,  0.78379792,
         0.8604924 ,  0.          ],
       [ 1.74590507,  0.43348119,
0.37103498, ..., -1.29836334,
         1.62022478,  0.          ],
       ...,
       [ 0.65211368,  0.43348119,
0.32994343, ...,  0.45065212,
         0.1482433 ,  0.          ],
       [ 0.65211368,  0.11634187,
0.80249633, ..., -1.04850399,
        -1.18128837,  0.          ],
       [-1.53546912, -1.46935473,
-1.24180861, ..., -1.88136849,
        -1.60863783,  0.          ]])
```

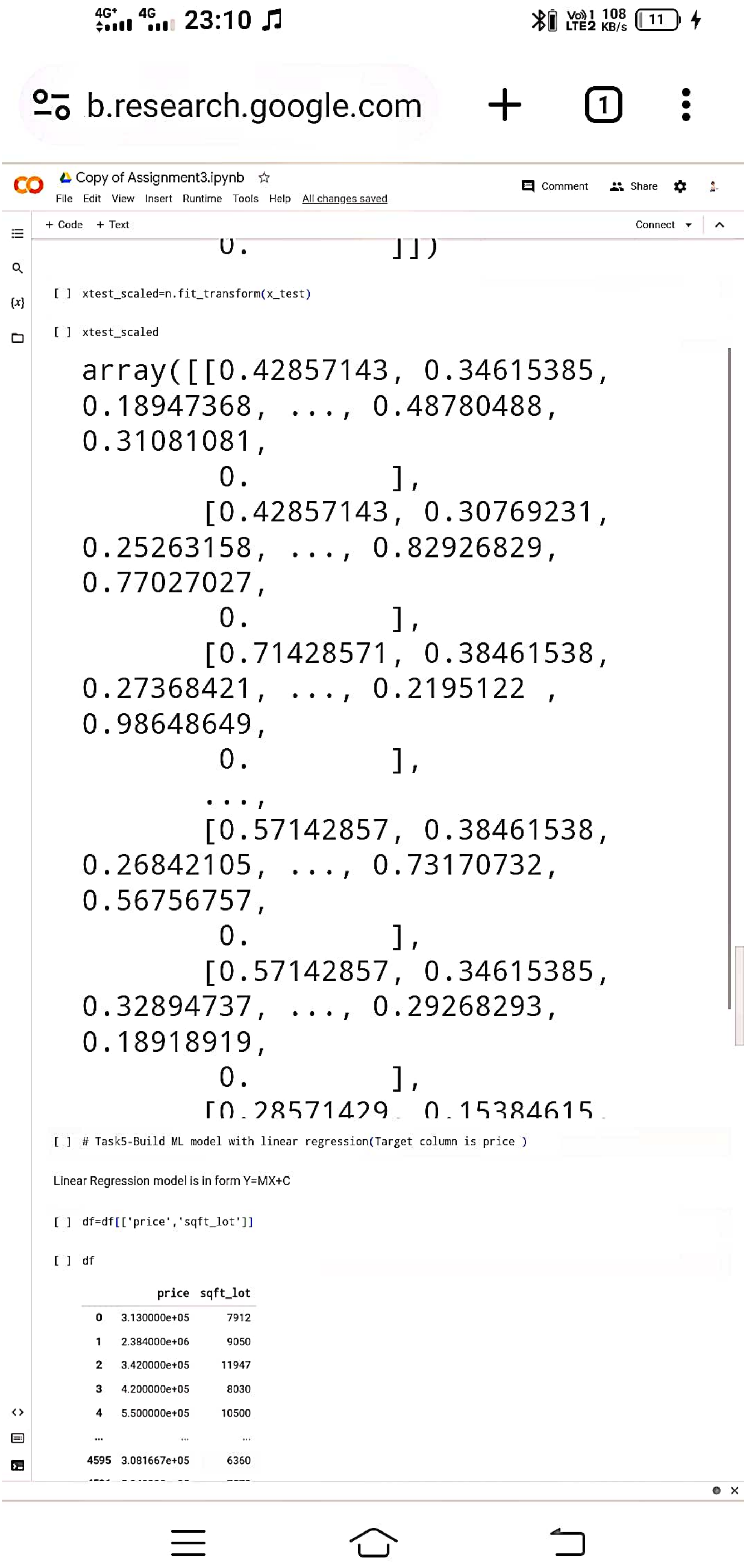
Min-Max Scaler

```
[ ] n = MinMaxScaler()
```

```
[ ] xtrain_scaled = n.fit_transform(x_train)
```

```
[ ] xtrain_scaled
```

```
array([[0.44444444, 0.125     ,
0.06985573, ..., 0.81395349,
0.85526316,
        0.        ],
       [0.22222222, 0.125     ,
0.02505695, ..., 0.81395349,
0.73684211,
        0.        ]
```


23:10
b.research.google.com
Copy of Assignment3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect

```
         0.          ]])
```

```
[ ] xtest_scaled=n.fit_transform(x_test)
```

```
[ ] xtest_scaled
```

```
array([[0.42857143, 0.34615385,
0.18947368, ..., 0.48780488,
0.31081081,
         0.          ],
        [0.42857143, 0.30769231,
0.25263158, ..., 0.82926829,
0.77027027,
         0.          ],
        [0.71428571, 0.38461538,
0.27368421, ..., 0.2195122 ,
0.98648649,
         0.          ],
        ...,
        [0.57142857, 0.38461538,
0.26842105, ..., 0.73170732,
0.56756757,
         0.          ],
        [0.57142857, 0.34615385,
0.32894737, ..., 0.29268293,
0.18918919,
         0.          ],
        [0.28571429, 0.15384615,
```

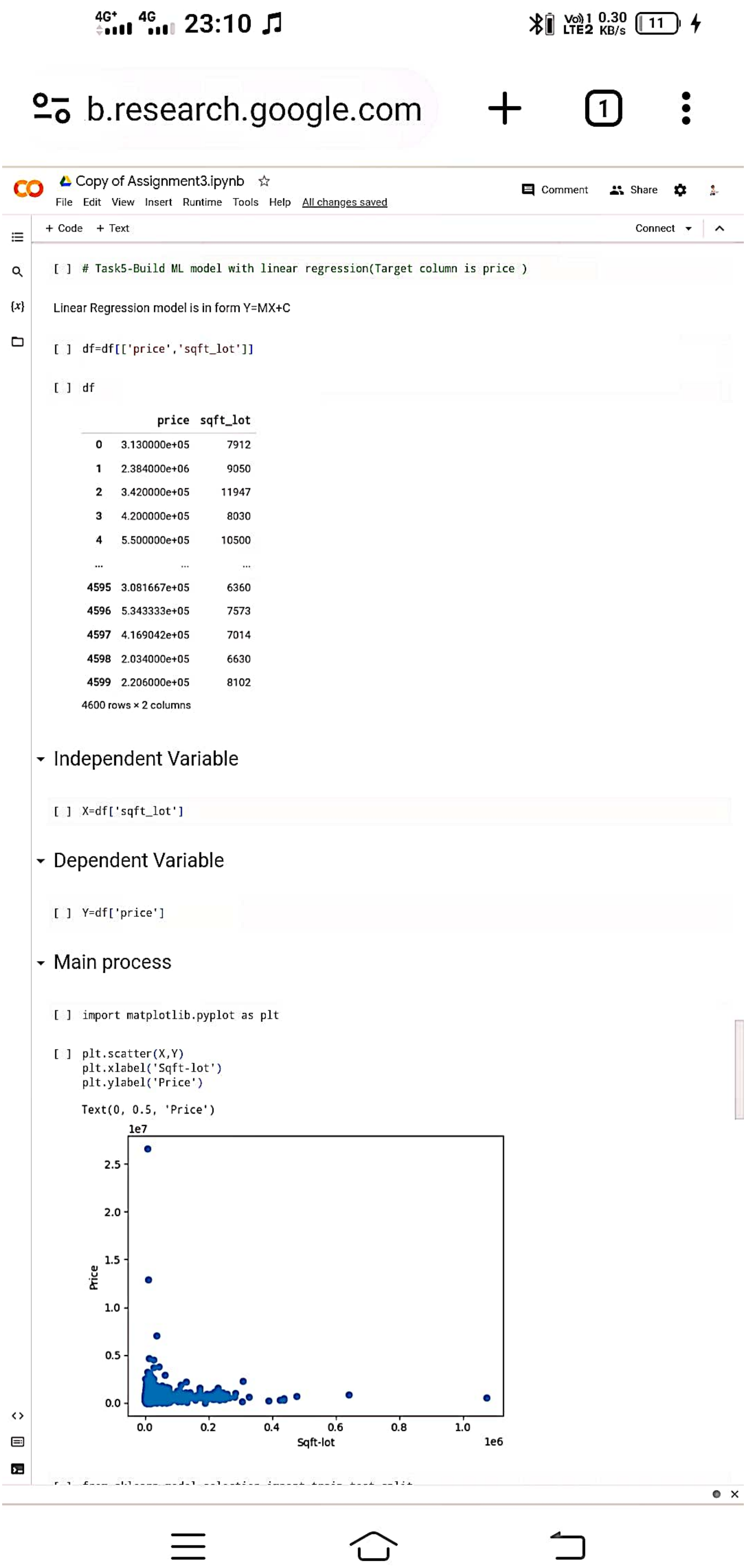
```
[ ] # Task5-Build ML model with linear regression(Target column is price )
```

Linear Regression model is in form Y=MX+C

```
[ ] df=df[['price','sqft_lot']]
```

```
[ ] df
```

	price	sqft_lot
0	3.130000e+05	7912
1	2.384000e+06	9050
2	3.420000e+05	11947
3	4.200000e+05	8030
4	5.500000e+05	10500
...	...	...
4595	3.081667e+05	6360


4G+ 4G 23:10
0.30 KB/s 11
b.research.google.com
1
CO
Copy of Assignment3.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect

```
[ ] # Task5-Build ML model with linear regression(Target column is price )
```

Linear Regression model is in form Y=MX+C

```
[ ] df=df[['price','sqft_lot']]
```

```
[ ] df
```

	price	sqft_lot
0	3.130000e+05	7912
1	2.384000e+06	9050
2	3.420000e+05	11947
3	4.200000e+05	8030
4	5.500000e+05	10500
...	...	...
4595	3.081667e+05	6360
4596	5.343333e+05	7573
4597	4.169042e+05	7014
4598	2.034000e+05	6630
4599	2.206000e+05	8102

4600 rows × 2 columns

Independent Variable

```
[ ] X=df['sqft_lot']
```

Dependent Variable

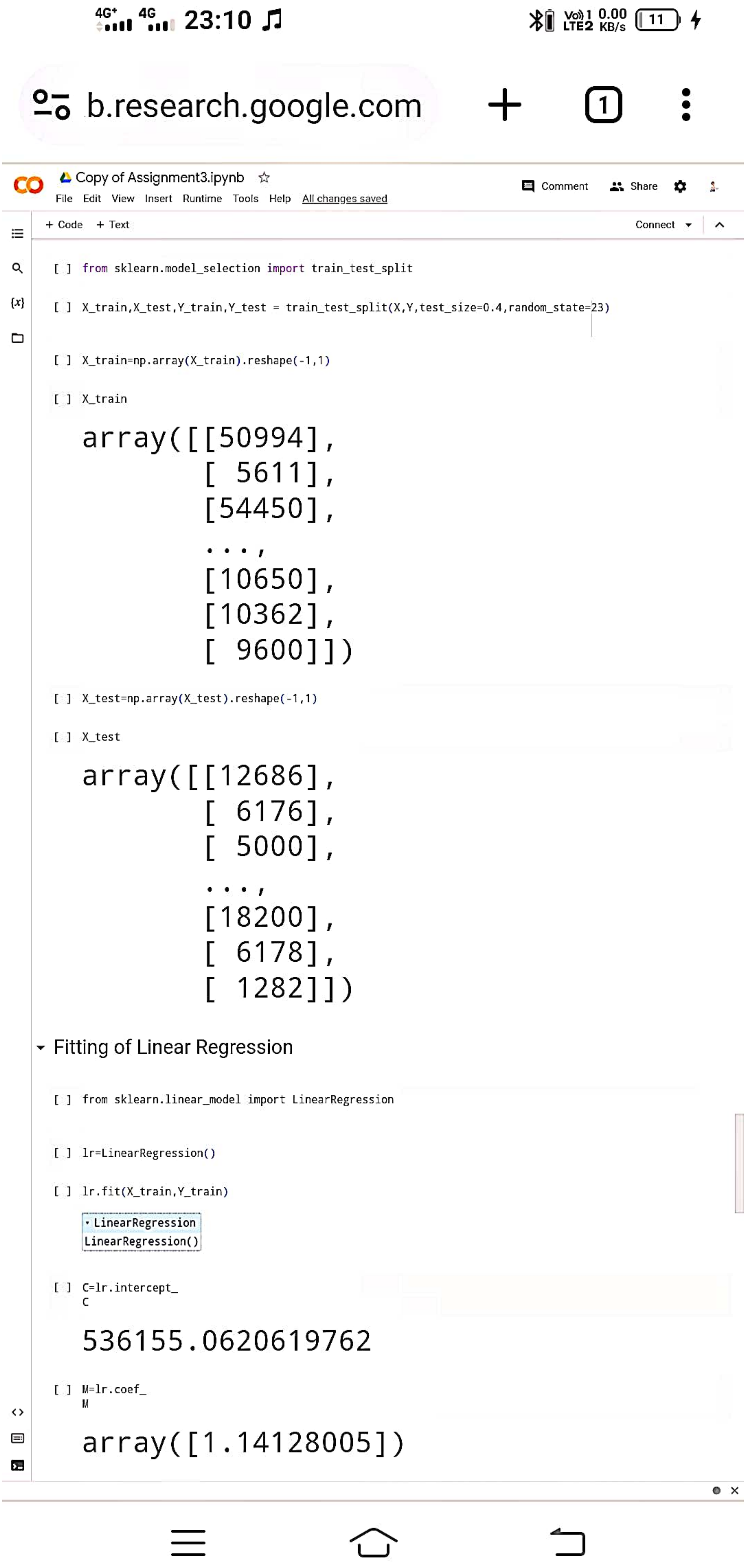
```
[ ] Y=df['price']
```

Main process

```
[ ] import matplotlib.pyplot as plt
```

```
[ ] plt.scatter(X,Y)
    plt.xlabel('Sqft-lot')
    plt.ylabel('Price')

    Text(0, 0.5, 'Price')
```

```
[ ] from sklearn.model_selection import train_test_split
```



```
[ ] from sklearn.model_selection import train_test_split
```

```
[ ] X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.4,random_state=23)
```

```
[ ] X_train=np.array(X_train).reshape(-1,1)
```

```
[ ] X_train

array([[50994],
       [ 5611],
       [54450],
       ...,
       [10650],
       [10362],
       [ 9600]])
```

```
[ ] X_test=np.array(X_test).reshape(-1,1)
```

```
[ ] X_test

array([[12686],
       [ 6176],
       [ 5000],
       ...,
       [18200],
       [ 6178],
       [ 1282]])
```

Fitting of Linear Regression

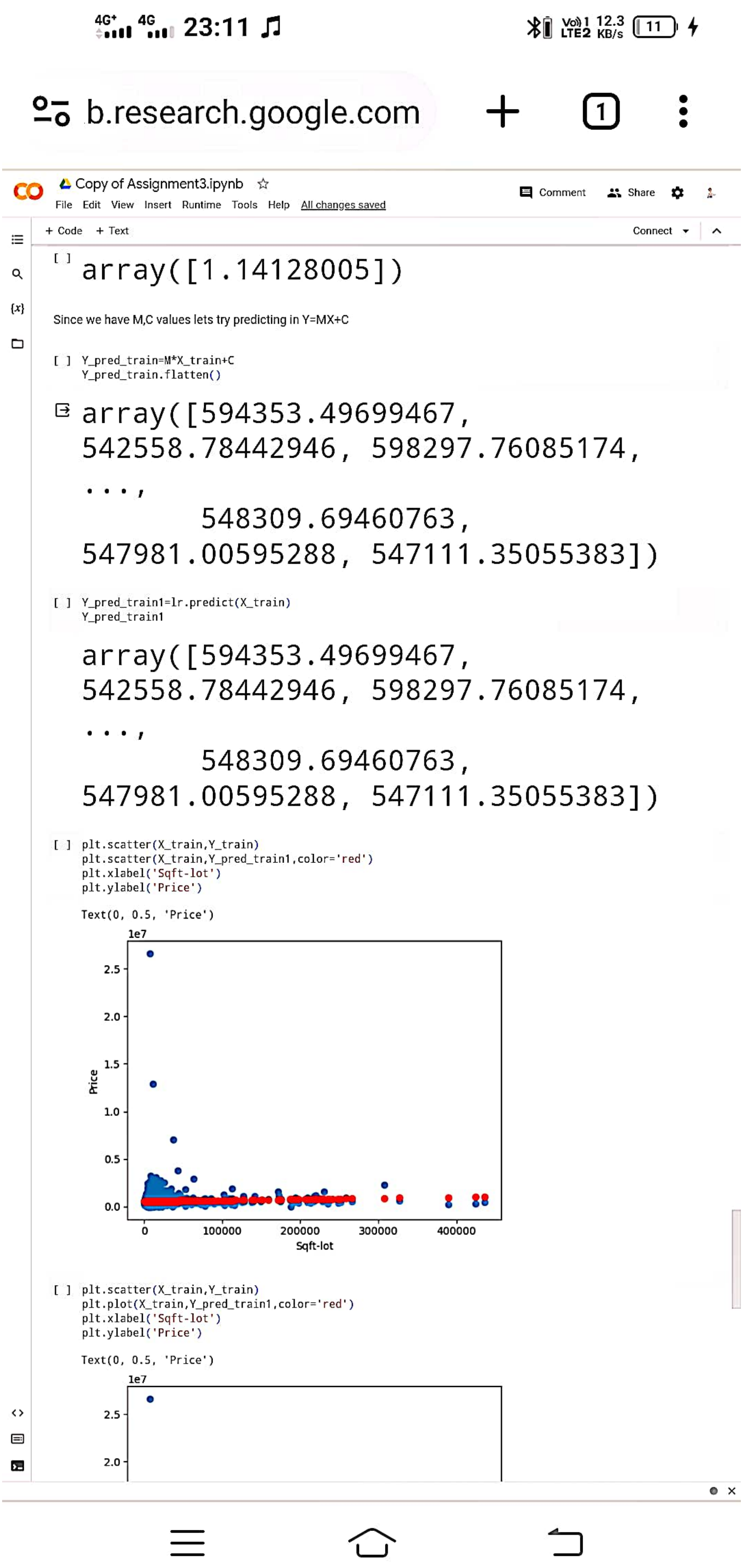
```
[ ] from sklearn.linear_model import LinearRegression
```

```
[ ] lr=LinearRegression()
```

```
[ ] lr.fit(X_train,Y_train)

▸ LinearRegression
LinearRegression()
```

```
[ ] C=lr.intercept_
    C

536155.0620619762
```

```
[ ] M=lr.coef_
    M

array([1.14128005])
```


4G+ 4G 23:11
12.3 KB/s 11
b.research.google.com
1
CO
Copy of Assignment3.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect

```
[ ] array([1.14128005])
```

Since we have M,C values lets try predicting in Y=MX+C

```
[ ] Y_pred_train=M*X_train+C
    Y_pred_train.flatten()
```

```
array([594353.49699467,
542558.78442946, 598297.76085174,
...,
        548309.69460763,
547981.00595288, 547111.35055383])
```

```
[ ] Y_pred_train1=lr.predict(X_train)
    Y_pred_train1
```

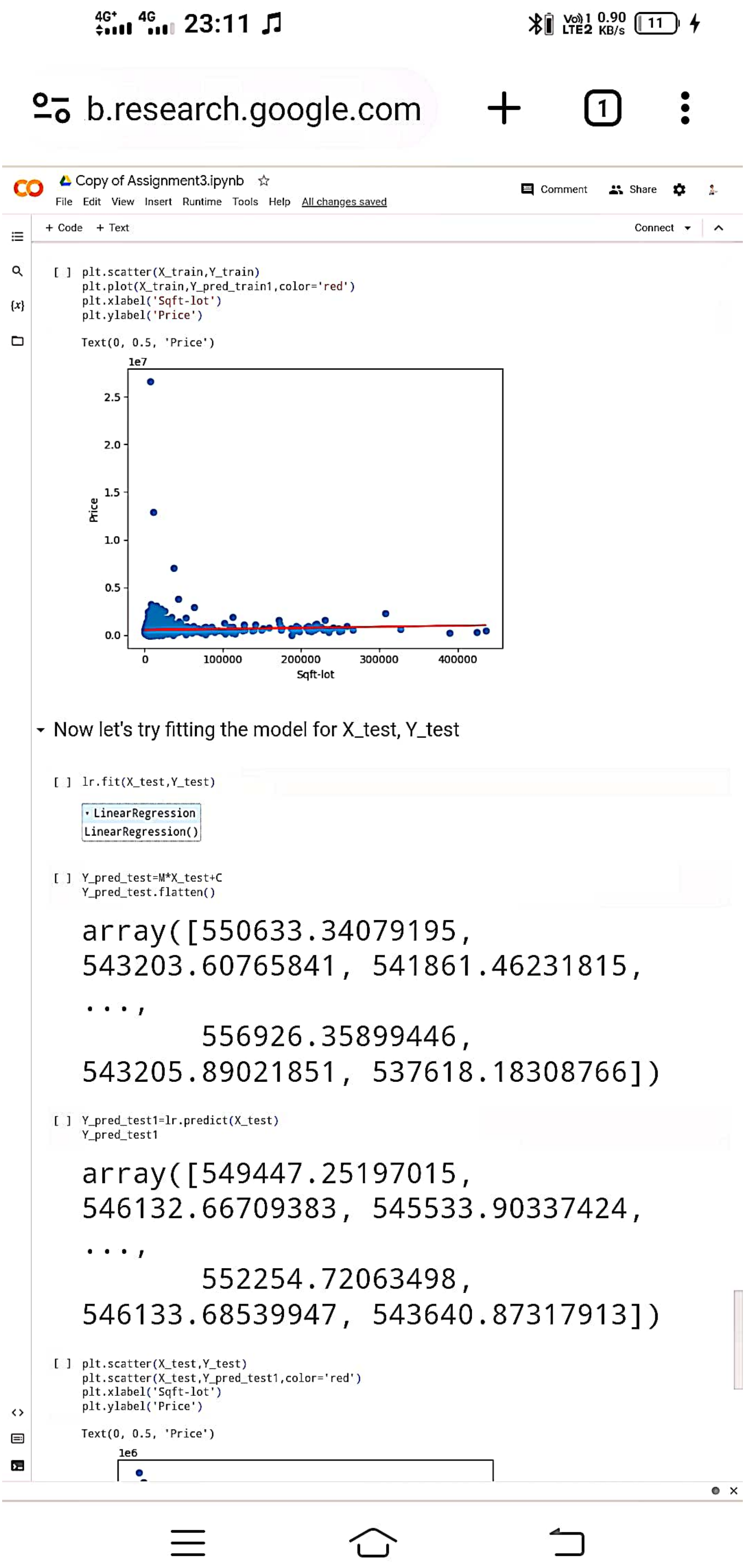
```
array([594353.49699467,
542558.78442946, 598297.76085174,
...,
        548309.69460763,
547981.00595288, 547111.35055383])
```

```
[ ] plt.scatter(X_train,Y_train)
    plt.scatter(X_train,Y_pred_train1,color='red')
    plt.xlabel('Sqft-lot')
    plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```

```
[ ] plt.scatter(X_train,Y_train)
    plt.plot(X_train,Y_pred_train1,color='red')
    plt.xlabel('Sqft-lot')
    plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```



```
[ ] plt.scatter(X_train,Y_train)
    plt.plot(X_train,Y_pred_train1,color='red')
    plt.xlabel('Sqft-lot')
    plt.ylabel('Price')

    Text(0, 0.5, 'Price')
```

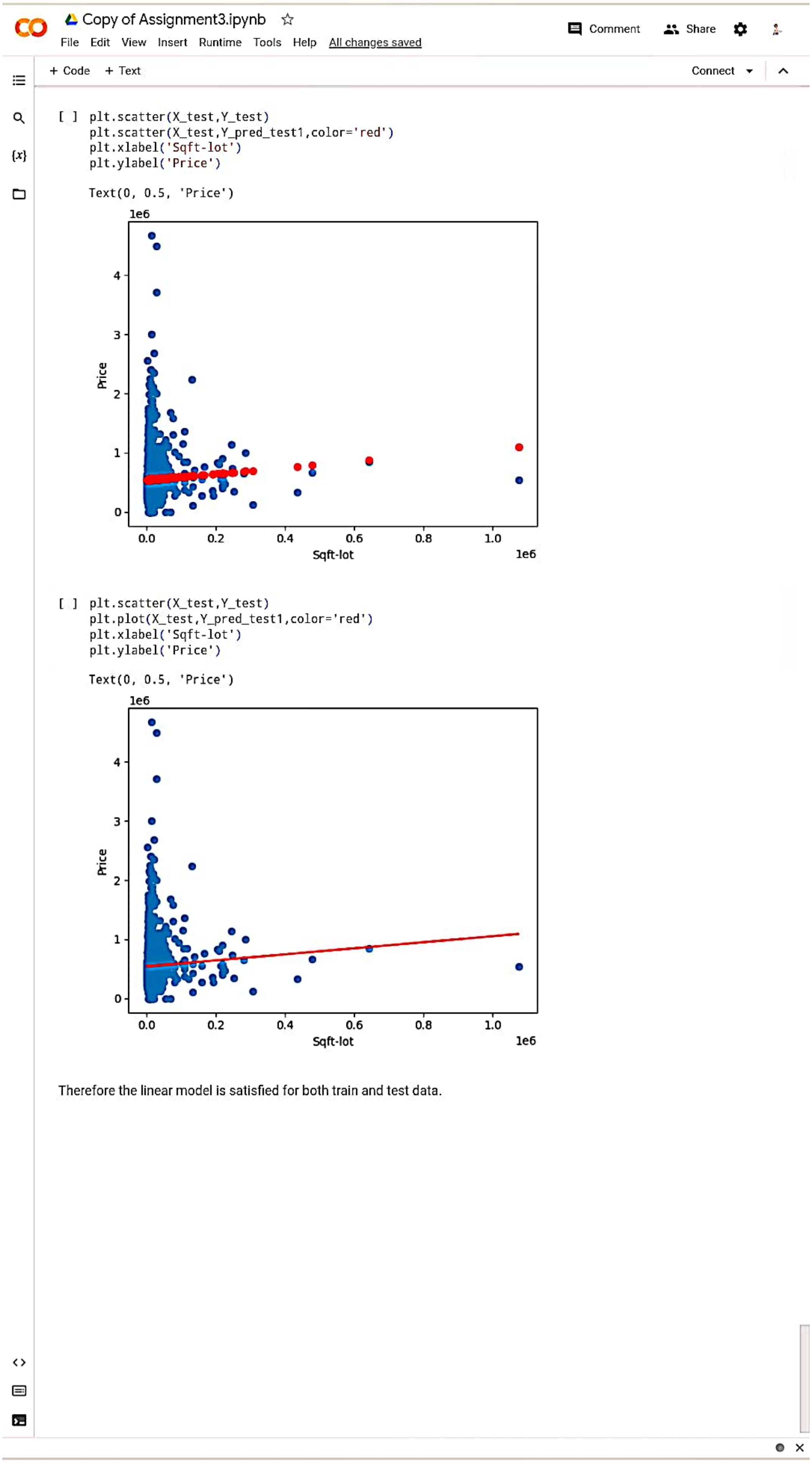
Now let's try fitting the model for X_test, Y_test

```
[ ] lr.fit(X_test,Y_test)

    ▸ LinearRegression
    LinearRegression()
```

```
[ ] Y_pred_test=M*X_test+C
    Y_pred_test.flatten()

    array([550633.34079195,
    543203.60765841, 541861.46231815,
    ...,
                556926.35899446,
    543205.89021851, 537618.18308766])
```

```
[ ] Y_pred_test1=lr.predict(X_test)
    Y_pred_test1

    array([549447.25197015,
    546132.66709383, 545533.90337424,
    ...,
                552254.72063498,
    546133.68539947, 543640.87317913])
```

```
[ ] plt.scatter(X_test,Y_test)
    plt.scatter(X_test,Y_pred_test1,color='red')
    plt.xlabel('Sqft-lot')
    plt.ylabel('Price')

    Text(0, 0.5, 'Price')
```


b.research.google.com
Copy of Assignment3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code
+ Text
Connect

```
[ ] plt.scatter(X_test,Y_test)
    plt.scatter(X_test,Y_pred_test1,color='red')
    plt.xlabel('Sqft-lot')
    plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```

```
[ ] plt.scatter(X_test,Y_test)
    plt.plot(X_test,Y_pred_test1,color='red')
    plt.xlabel('Sqft-lot')
    plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```

Therefore the linear model is satisfied for both train and test data.