U18ISI6204 – Machine Learning Techniques
LAB- EXPERIMENT 2

NAME: Aaron Mathew
ROLL_NO: 20BIS001

OBJECTIVE OF THE EXERCISE/EXPERIMENT

To perform Linear regression in single and multiple variables on the given dataset, using scikit library

STEP-1: Start the program.

STEP-2: import all the necessary libraries

i) Numpy – array manipulation

ii) Pandas – dataframe manipulation

iii) Matplotlib and seaborn – for data visualization

iv) Sklearn.model_selection – train test data split

v) Sklearn.metrics – mean square error and r2 score.

vi) Sklearn,linear_model – for linear regression

STEP-3: Loading the dataset using read_csv method in pandas module.

STEP-4: Analyze the dataset using info method, which gives its data types and number of non- null values in each columns.

STEP-5: Perform basic statistic operation using describe() method.

STEP-6: Use heatmaps, correlation matrix, regression plots and pairplots in seaborn to find the relationship between features.

STEP-7: Implement Simple Linear regression(singleLR) with only one variable

(X3 distance to the nearest MRT station) and calculate the MSE and R2 score for the singleLR model.

STEP-8: Implement Multiple Linear regression(multiLR) with selected variable (refined cols) which are pick out by analyzing the relationship between features and calculate the MSE and R2 score for the multiLR model.

STEP-9: Stop the program.

PRICE PREDICTION DATASET

SYNTAX:

import pandas as pd
data=pd.read_csv("C:/Users/MADL22/Downloads/archive/data.csv")
Data

```
In [7]: data
```

Out[7]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	2005
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	0
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	0
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	0
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976	1992
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	1510	0	1954	1979
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	1460	0	1983	2009

CORRELATION:

SYNTAX:

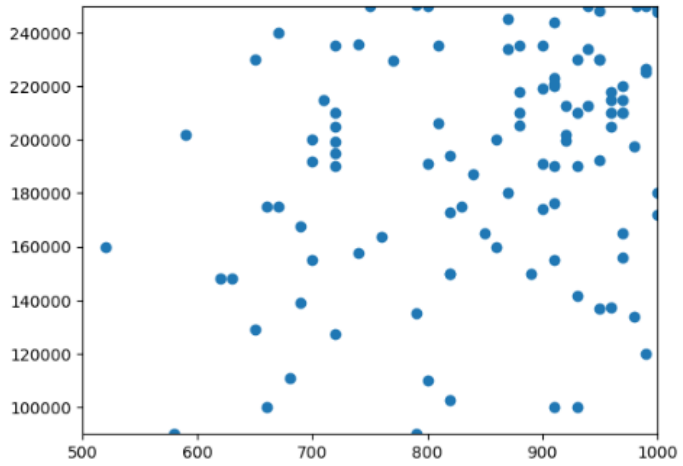
cor=data['price'].corr(data['sqft_lot'])
Cor

```
In [15]: cor=data['price'].corr(data['sqft_lot'])

In [16]: cor
Out[16]: 0.05045129503234886

In [ ]:
```

REGRESSION:
SYNTAX:

```
import matplotlib.pyplot as plt
from scipy import stats
x=data['sqft_living']
y=data['price']
slope, intercept, r, p, std_err = stats.linregress(x, y)
plt.xlim(500,1000)
plt.ylim(90000,250000)
plt.scatter(x,y)
```

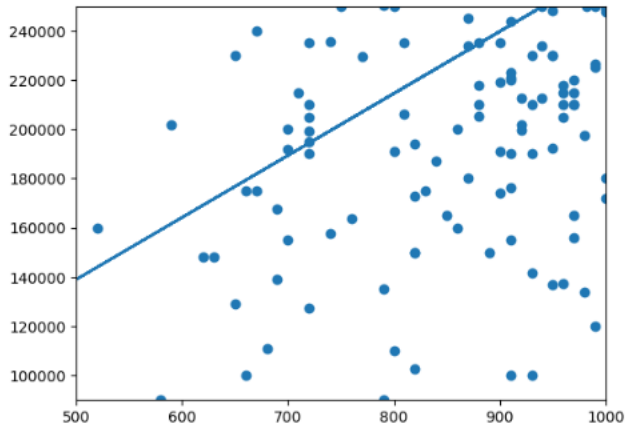
```
In [71]: plt.xlim(500,1000)
         plt.ylim(90000,250000)
         plt.scatter(x,y)

Out[71]: <matplotlib.collections.PathCollection at 0x1667adfba90>
```



SLOPE EQUATION:
SYNTAX:

```
def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))
plt.xlim(500,1000)
plt.ylim(90000,250000)
plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```

```
In [70]: def myfunc(x):
           return slope * x + intercept

         mymodel = list(map(myfunc, x))
         plt.xlim(500,1000)
         plt.ylim(90000,250000)
         plt.scatter(x, y)
         plt.plot(x, mymodel)
         plt.show()
```



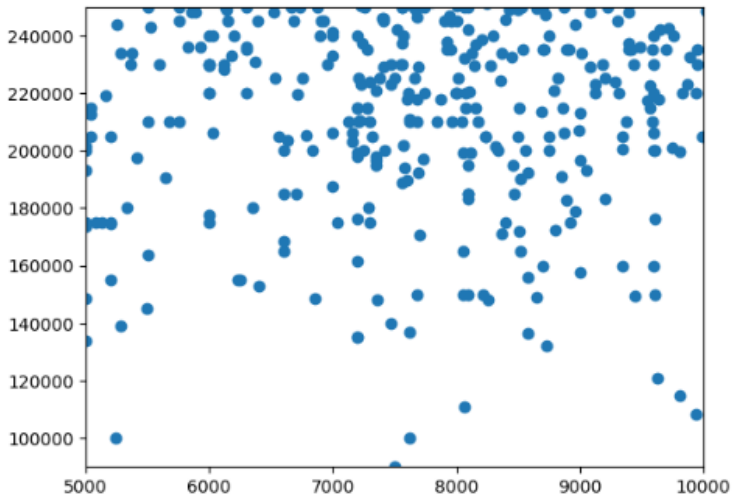

```
In [ ]:
```

GRAPH 2:

SYNTAX:

x=data['sqft_lot']
y=data['price']
slope, intercept, r, p, std_err = stats.linregress(x, y)
plt.xlim(5000,10000)
plt.ylim(90000,250000)
plt.scatter(x,y)

```
In [74]: x=data['sqft_lot']
         y=data['price']
         slope, intercept, r, p, std_err = stats.linregress(x, y)
         plt.xlim(5000,10000)
         plt.ylim(90000,250000)
         plt.scatter(x,y)

Out[74]: <matplotlib.collections.PathCollection at 0x1667bee1340>
```



SLOPE:

```
def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))
plt.xlim(300000,100000)
plt.ylim(100000,2500000)
plt.scatter(x, y)
plt.plot(x, mymodel)

plt.show()
```

In [82]:

```
def myfunc(x):
  return slope * x + intercept

mymodel = list(map(myfunc, x))
plt.xlim(300000,100000)
plt.ylim(100000,2500000)
plt.scatter(x, y)
plt.plot(x, mymodel)

plt.show()
```