- **Nama : Davit Cany Agho**
- **Nim : G.211.21.0116**

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

from google.colab import files
```

```
dataset = pd.read_csv('kc_house_data.csv')
```

```
Y = dataset[['price']]
```

```
X = dataset.drop(['price', 'id', 'date'], axis=1)
```

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   bedrooms       21613 non-null  int64
 1   bathrooms      21613 non-null  float64
 2   sqft_living    21613 non-null  int64
 3   sqft_lot       21613 non-null  int64
 4   floors         21613 non-null  float64
 5   waterfront     21613 non-null  int64
 6   view           21613 non-null  int64
 7   condition      21613 non-null  int64
 8   grade          21613 non-null  int64
 9   sqft_above     21613 non-null  int64
 10  sqft_basement  21613 non-null  int64
 11  yr_built       21613 non-null  int64
 12  yr_renovated   21613 non-null  int64
 13  zipcode        21613 non-null  int64
 14  lat            21613 non-null  float64
 15  long           21613 non-null  float64
 16  sqft_living15  21613 non-null  int64
 17  sqft_lot15     21613 non-null  int64
dtypes: float64(4), int64(14)
memory usage: 3.0 MB
```

```
columns = X.columns
columns
```

```
Index(['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors',
       'waterfront', 'view', 'condition', 'grade', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode', 'lat', 'long',
       'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

```
X.head()
```

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	y
0	3	1.00	1180	5650	1.0	0	0	3	7	1180	0	1955	
1	3	2.25	2570	7242	2.0	0	0	3	7	2170	400	1951	
2	2	1.00	770	10000	1.0	0	0	3	6	770	0	1933	
3	4	3.00	1960	5000	1.0	0	0	5	7	1050	910	1965	
4	3	2.00	1680	8080	1.0	0	0	3	8	1680	0	1987	

```
X.describe()
```

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade
count	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000
mean	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.656873
std	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.175459
min	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	1.000000
25%	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.000000
50%	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.000000
75%	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000	8.000000
max	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.000000

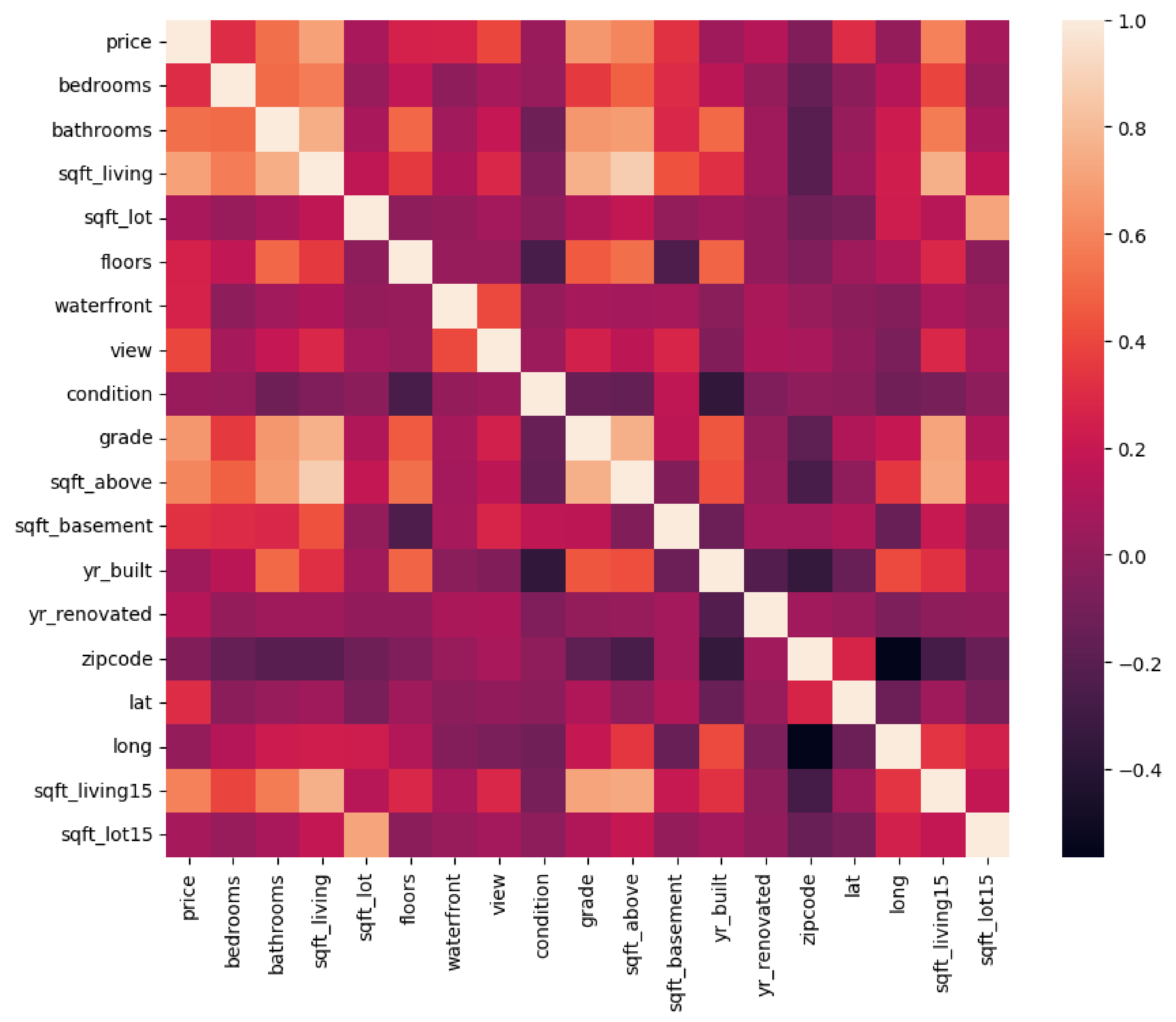
```
dataset = dataset.drop(['id', 'date'], axis=1)
```

```
dataset.corr(method='pearson')
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_a
price	1.000000	0.308350	0.525138	0.702035	0.089661	0.256794	0.266369	0.397293	0.036362	0.667434	0.60
bedrooms	0.308350	1.000000	0.515884	0.576671	0.031703	0.175429	-0.006582	0.079532	0.028472	0.356967	0.47
bathrooms	0.525138	0.515884	1.000000	0.754665	0.087740	0.500653	0.063744	0.187737	-0.124982	0.664983	0.68
sqft_living	0.702035	0.576671	0.754665	1.000000	0.172826	0.353949	0.103818	0.284611	-0.058753	0.762704	0.87
sqft_lot	0.089661	0.031703	0.087740	0.172826	1.000000	-0.005201	0.021604	0.074710	-0.008958	0.113621	0.18
floors	0.256794	0.175429	0.500653	0.353949	-0.005201	1.000000	0.023698	0.029444	-0.263768	0.458183	0.52
waterfront	0.266369	-0.006582	0.063744	0.103818	0.021604	0.023698	1.000000	0.401857	0.016653	0.082775	0.07
view	0.397293	0.079532	0.187737	0.284611	0.074710	0.029444	0.401857	1.000000	0.045990	0.251321	0.16
condition	0.036362	0.028472	-0.124982	-0.058753	-0.008958	-0.263768	0.016653	0.045990	1.000000	-0.144674	-0.15
grade	0.667434	0.356967	0.664983	0.762704	0.113621	0.458183	0.082775	0.251321	-0.144674	1.000000	0.75
sqft_above	0.605567	0.477600	0.685342	0.876597	0.183512	0.523885	0.072075	0.167649	-0.158214	0.755923	1.00
sqft_basement	0.323816	0.303093	0.283770	0.435043	0.015286	-0.245705	0.080588	0.276947	0.174105	0.168392	-0.05
yr_built	0.054012	0.154178	0.506019	0.318049	0.053080	0.489319	-0.026161	-0.053440	-0.361417	0.446963	0.42
yr_renovated	0.126434	0.018841	0.050739	0.055363	0.007644	0.006338	0.092885	0.103917	-0.060618	0.014414	0.02
zipcode	-0.053203	-0.152668	-0.203866	-0.199430	-0.129574	-0.059121	0.030285	0.084827	0.003026	-0.184862	-0.26
lat	0.307003	-0.008931	0.024573	0.052529	-0.085683	0.049614	-0.014274	0.006157	-0.014941	0.114084	-0.00
long	0.021626	0.129473	0.223042	0.240223	0.229521	0.125419	-0.041910	-0.078400	-0.106500	0.198372	0.34
sqft_living15	0.585379	0.391638	0.568634	0.756420	0.144608	0.279885	0.086463	0.280439	-0.092824	0.713202	0.73
sqft_lot15	0.082447	0.029244	0.087175	0.183286	0.718557	-0.011269	0.030703	0.072575	-0.003406	0.119248	0.19

```
plt.subplots(figsize=(10,8))
sns.heatmap(dataset.corr())
```

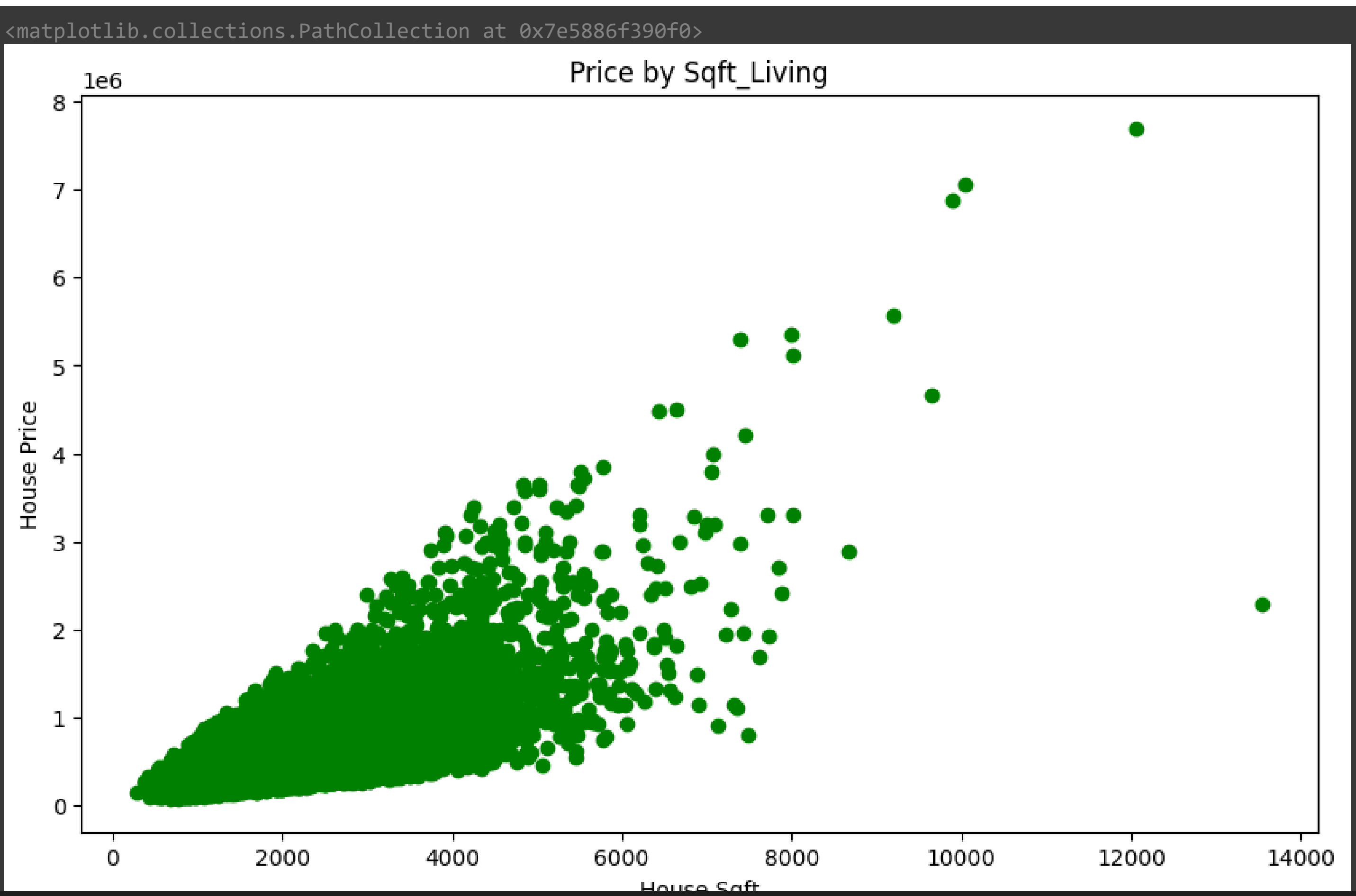
```
<Axes: >
```



```
x= X[['sqft_living']]
y = Y
```

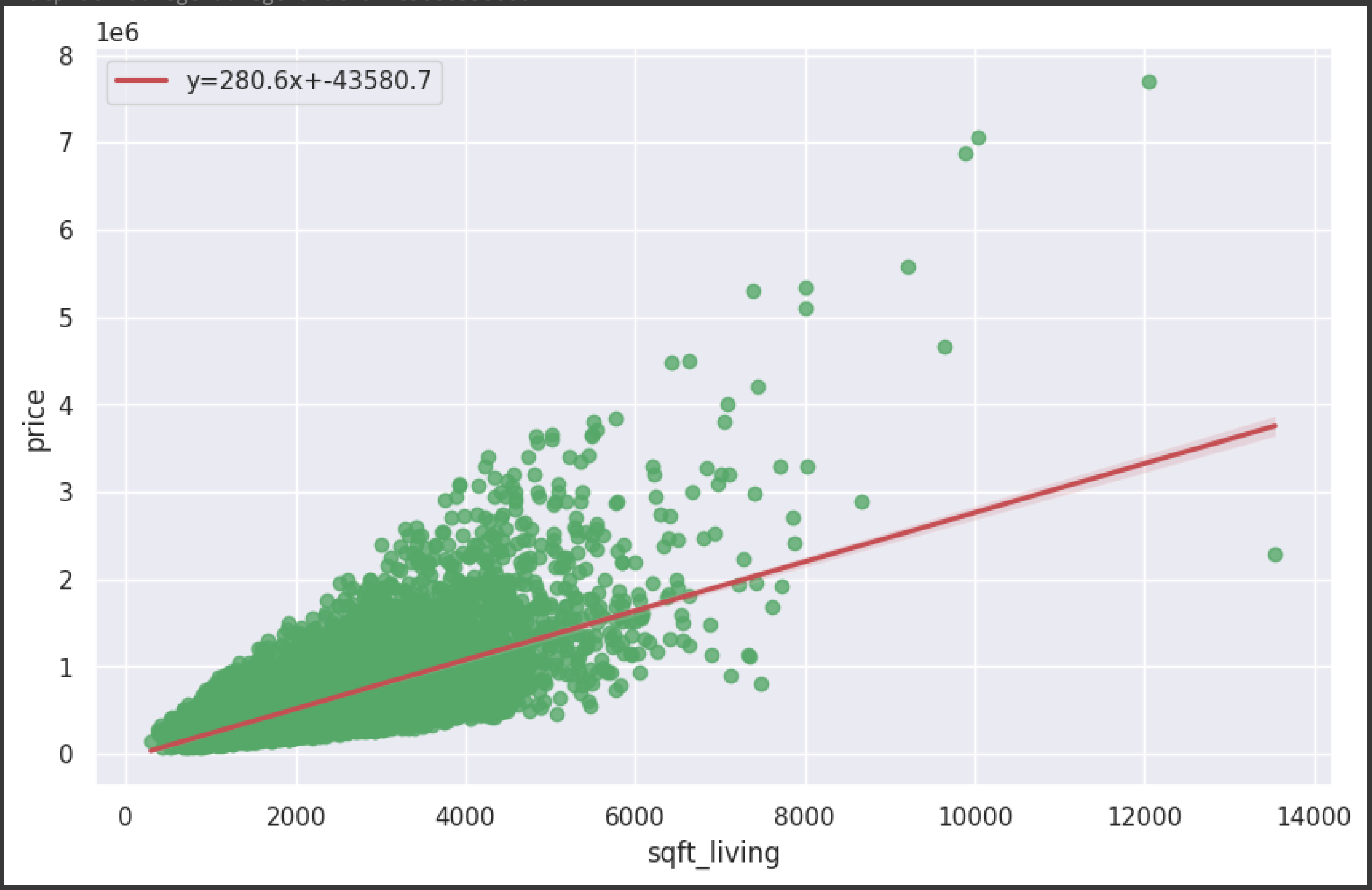
```
plt.figure(figsize=(10,6))
plt.xlabel('House Sqft')
plt.ylabel('House Price')
plt.title('Price by Sqft_Living')
plt.scatter(x,y, marker='o', color='g')
```

```
<matplotlib.collections.PathCollection at 0x7e5886f390f0>
```



```
from scipy import stats
sns.set(color_codes=True)

slope, intercept, r_value, p_value, std_err = stats.linregress(dataset['sqft_living'],dataset['price'])

f = plt.figure(figsize=(10,6))
data = dataset[['price', 'sqft_living']]
ax = sns.regplot(x='sqft_living', y='price', data=data,
                 scatter_kws={"color": "g"},
                 line_kws={'color': 'r', 'label':"y={0:.1f}x+{1:.1f}".format(slope,intercept)})
ax.legend()
```

```
<matplotlib.legend.Legend at 0x7e588633b880>
```



```
x = X[['sqft_living']]
y = Y
```

```
xg = x.values.reshape(-1,1)
yg = y.values.reshape(-1,1)
xg = np.concatenate((np.ones(len(x)).reshape(-1,1), x), axis=1)
```

```
def computeCost(x, y, theta):
  m= len(y)
  h_x = x.dot(theta)
  j = np.sum(np.square(h_x - y))*(1/(2*m))
  return j
```

```
def gradientDescent(x, y, theta, alpha, iteration):
  print("Running Gradient Descent...")
  j_hist = []
  m = len(y)
  for i in range(iteration):
    j_hist.append(computeCost(x, y, theta))
    h_x = x.dot(theta)
    theta = theta - ((alpha/m) *((np.dot(x.T, (h_x-y) ))))
  return theta, j_hist
```

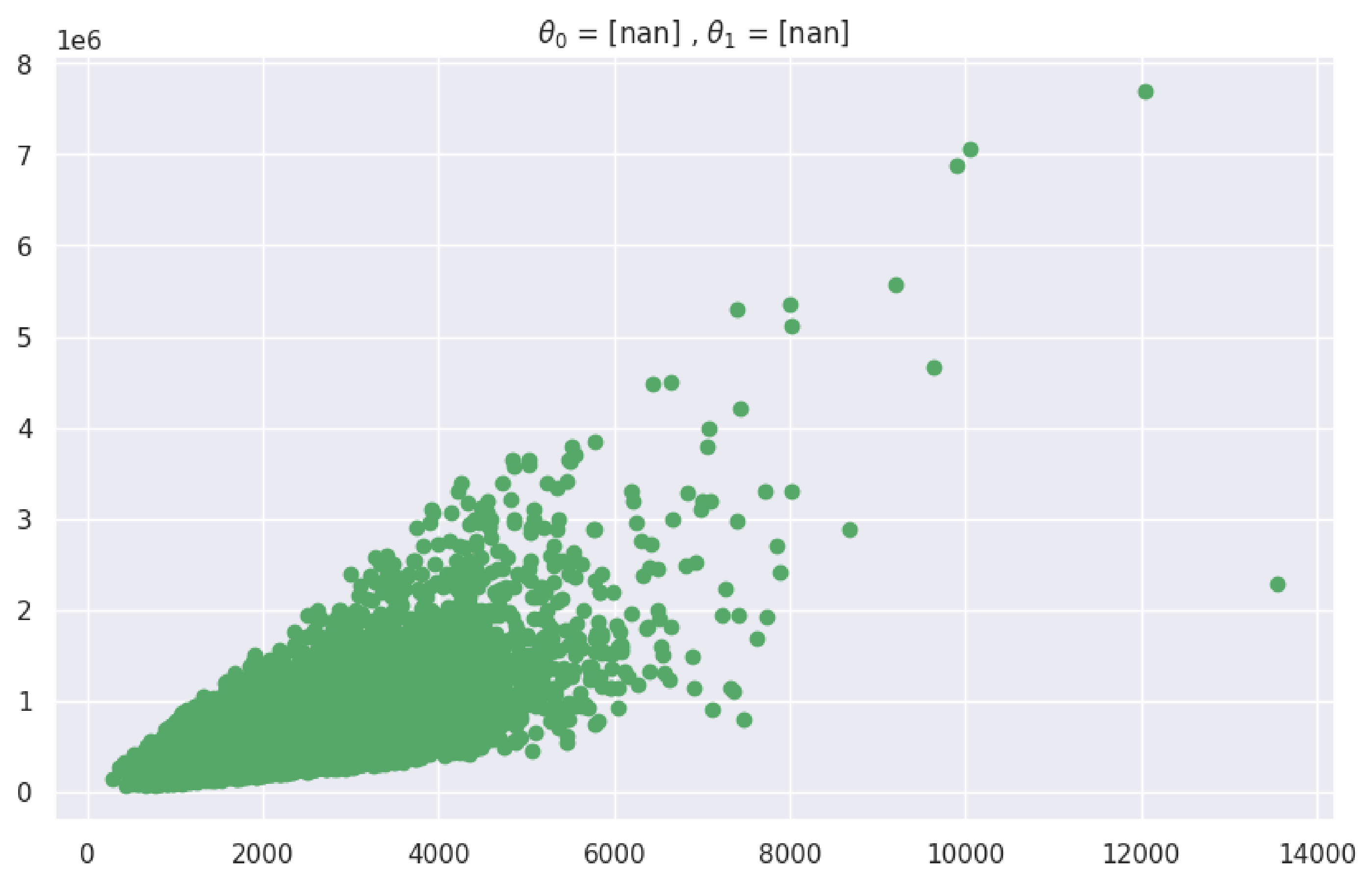
```
theta = np.zeros((2,1))
iteration = 2000
alpha = 0.001

theta, cost = gradientDescent(xg, yg, theta, alpha, iteration)
print("Thetha found by Gradient Descent: slope = {} and itercept {}".format(theta[1], theta[0]))
```

```
Running Gradient Descent...
<ipython-input-42-a0e29db2a29a>:4: RuntimeWarning: overflow encountered in square
  j = np.sum(np.square(h_x - y))*(1/(2*m))
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:86: RuntimeWarning: overflow encountered in reduce
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
<ipython-input-43-dce3ed1aeb91>:8: RuntimeWarning: invalid value encountered in subtract
  theta = theta - ((alpha/m) *((np.dot(x.T, (h_x-y) ))))
Thetha found by Gradient Descent: slope = [nan] and itercept [nan]
```

```
theta.shape
```

```
(2, 1)
```

```
plt.figure(figsize=(10,6))
plt.title('$\\theta_0$ = {} , $\\theta_1$ = {}'.format(theta[0], theta[1]))
plt.scatter(x,y, marker='o', color='g')
plt.plot(x,np.dot(x.values, theta.T))
plt.show()
```



```
plt.plot(cost)
plt.xlabel('No, of iterations')
plt.ylabel('Cost')
```

```
Text(0, 0.5, 'Cost')
1e300
```

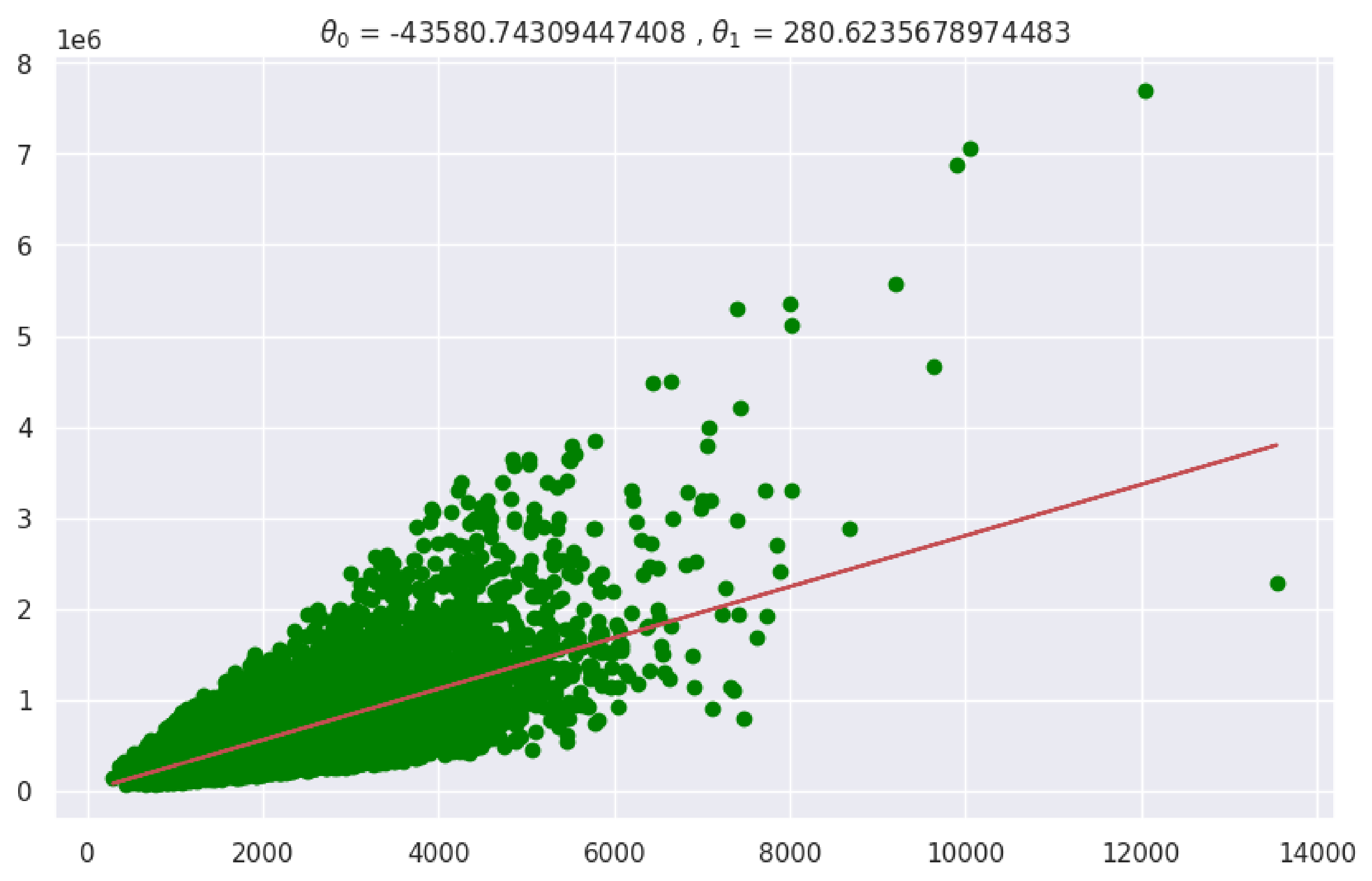
```
from scipy import stats

xs = x.iloc[:,0]
ys = y.iloc[:,0]

slope, intercept, r_value, p_value, std_err = stats.linregress(xs, ys)
```

```
print('Slope = {} and Intercept = {}'.format(slope, intercept))
print('y = x({}) + {}'.format(slope, intercept))
```

```
Slope = 280.6235678974483 and Intercept = -43580.74309447408
y = x(280.6235678974483) + -43580.74309447408
```

```
plt.figure(figsize=(10,6))
plt.title('$\\theta_0$ = {} , $\\theta_1$ = {}'.format(intercept, slope))
plt.scatter(xs,y, marker='o', color='green')
plt.plot(xs, np.dot(x, slope), 'r')
```

```
[<matplotlib.lines.Line2D at 0x7e587f612440>]
```

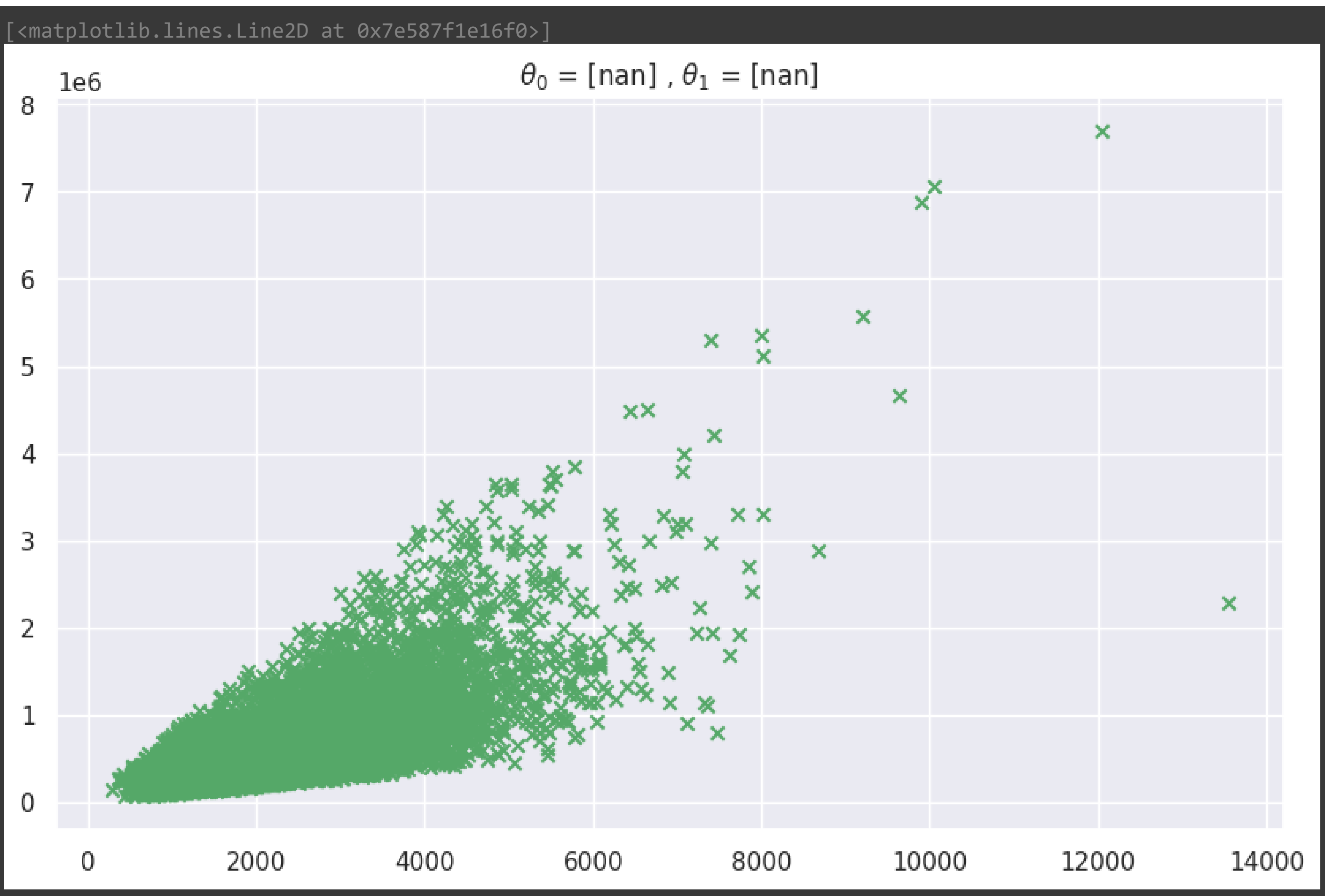


```
xsl = x.values.reshape(-1,1)
ysl = y.values.reshape(-1,1)
xsl = np.concatenate((np.ones(len(xsl)).reshape(-1,1), xsl), axis=1)

from sklearn.linear_model import LinearRegression

slr = LinearRegression()
slr.fit(xsl[:,1].reshape(-1,1), ysl.reshape(-1,1))
y_hat = slr.predict(xsl[:,1].reshape(-1,1))

print('theta[0] = ', slr.intercept_)
print('theta[1] = ', slr.coef_)

theta - np.array((slr.intercept_, slr.coef_)).squeeze()
```

```
theta[0] =  [-43580.74309447]
theta[1] =  [[280.6235679]]
<ipython-input-58-67e589bcb06b>:14: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tup
  theta - np.array((slr.intercept_, slr.coef_)).squeeze()
array([[nan, nan],
       [nan, nan]], dtype=object)
```

```
plt.figure(figsize=(10,6))
plt.title('$\\theta_0$ = {} , $\\theta_1$ = {}'.format(theta[0], theta[1]))
plt.scatter(xsl[:,1],y, marker='x', color='g')
plt.plot(xsl[:,1], np.dot(xsl, theta), 'r')
```

```
[<matplotlib.lines.Line2D at 0x7e587f1e16f0>]
```


θ_0 = [nan] , θ_1 = [nan]
1e6
8
7
6
5
4
3
2
1
0
0
2000
4000
6000
8000
10000
12000
14000