```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D


train=pd.read_csv('/content/crime_train.csv')
train=train.drop(["ID"],axis=1)
train.head()
```

	population	householdsize	agePct12t21	agePct12t29	agePct16t24	agePct65up	numbUrl
0	14985	2.56	16.55	34.42	22.54	10.13	
1	30843	2.83	15.45	35.12	18.14	4.70	
2	74991	2.52	10.48	20.43	9.11	20.68	73
3	45061	2.44	10.59	24.97	11.61	16.34	45
4	12863	2.45	12.02	22.51	10.49	18.46	

5 rows × 89 columns

```
train.describe()
```

	population	householdsize	agePct12t21	agePct12t29	agePct16t24	agePct65up
count	1.595000e+03	1595.000000	1595.000000	1595.000000	1595.000000	1595.000000
mean	5.403041e+04	2.702514	14.409141	27.593806	13.944846	11.959335
std	2.195193e+05	0.341554	4.434560	6.136254	5.883211	4.771171
min	1.000500e+04	1.810000	4.680000	9.380000	4.640000	1.660000
25%	1.437350e+04	2.490000	12.240000	24.375000	11.315000	8.985000
50%	2.292200e+04	2.640000	13.640000	26.730000	12.520000	11.830000
75%	4.423950e+04	2.840000	15.345000	29.120000	14.340000	14.470000
max	7.322564e+06	5.280000	54.400000	70.510000	63.620000	52.770000

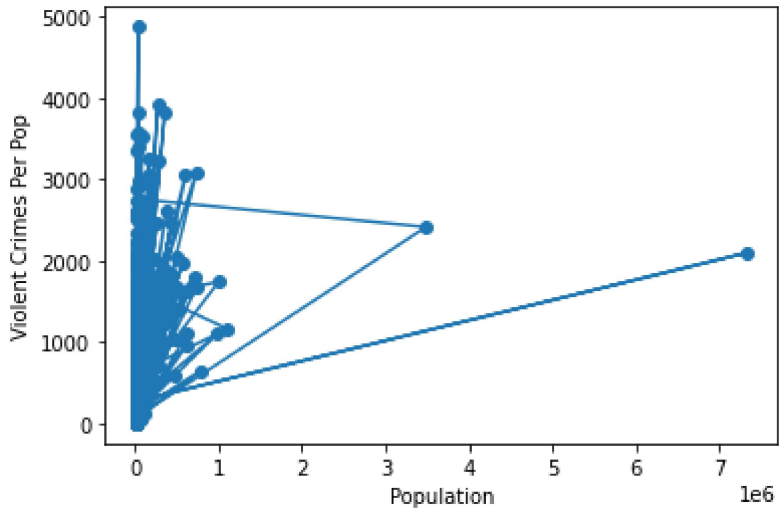
8 rows × 89 columns

```
test=pd.read_csv('/content/crime_test.csv')
test=test.drop(["ID"],axis=1)
test.head()
```

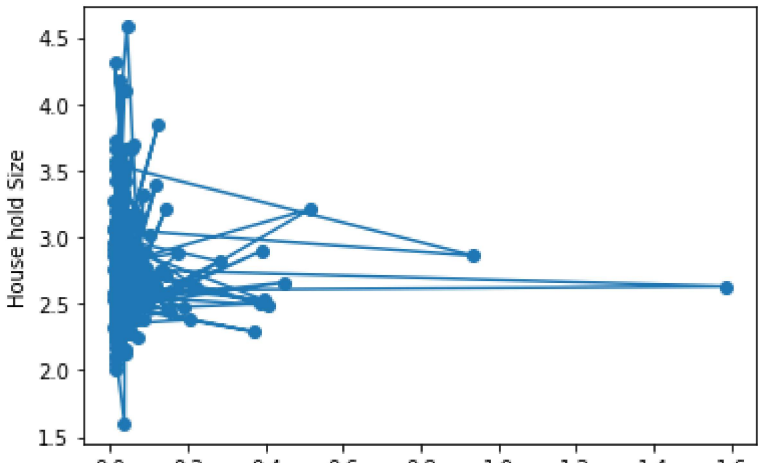
	population	householdsize	agePct12t21	agePct12t29	agePct16t24	agePct65up	numbUrl
0	11874	2.11	10.54	30.87	14.08	8.16	11
1	14143	2.68	21.01	33.35	21.95	14.55	
2	34882	2.32	12.56	21.79	11.29	19.51	34
3	29885	3.53	20.10	34.33	18.31	8.18	29
4	935933	2.86	15.89	30.35	14.98	9.50	935

5 rows × 88 columns

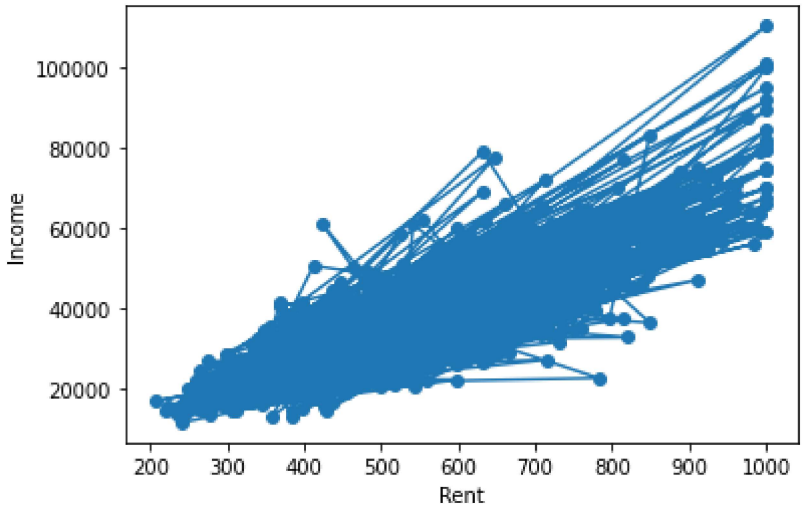
```
plt.scatter(train.population,train.ViolentCrimesPerPop)
plt.plot(train.population,train.ViolentCrimesPerPop)
plt.xlabel('Population')
plt.ylabel('Violent Crimes Per Pop')
plt.show()
```




```
plt.scatter(test.population,test.householdsize)
plt.plot(test.population,test.householdsize)
plt.xlabel('Population')
plt.ylabel('House hold Size')
plt.show()
```




```
plt.scatter(train.MedRent,train.medIncome)
plt.plot(train.MedRent,train.medIncome)
plt.xlabel('Rent')
plt.ylabel('Income')
plt.show()
```




```
from sklearn.linear_model import LinearRegression


X_T=train.iloc[:,:-1]
Y_T=train.iloc[:,-1]
reg=LinearRegression()
reg.fit(X_T,Y_T)
reg.score(X_T,Y_T)
reg.predict(X_T)
```

```
array([ 259.87557878,  567.1237    ,  413.71565939, ..., 1126.70024816,
        1056.1260383 ,  339.0973182 ])
```

```
X_T=test.iloc[:,:-1]
Y_T=test.iloc[:,-1]
reg=LinearRegression()
reg.fit(X_T,Y_T)
```

```
reg.score(X_T,Y_T)
reg.predict(X_T[0:20])
```

```
array([85.60254379, 89.9995928 , 90.81934825, 91.60551927, 90.82659769,
       75.90451551, 81.30849646, 89.18677298, 90.2972196 , 86.0989516 ,
       86.02252353, 90.80515326, 93.97619802, 78.51695206, 88.20714127,
       83.96128167, 95.14669323, 70.25666301, 89.43291127, 88.85064114])
```

```
from scipy import stats
slope, intercept, r, p, std_err = stats.linregress(train.population,train.ViolentCrimesPerPop
print('Slope = ',slope)
print('Intercept = ',slope)
def myfunc(x):
  return slope * x + intercept

print("Enter a random population to get predicted Violent Crime Per Pop")
pop=int(input())
crime = myfunc(pop)
print("When the population = ",pop," Violent Crime Per Pop = ", crime)
import math
from sklearn.metrics import mean_squared_error
rmse = math.sqrt(mean_squared_error([30843,742.54],[pop,crime]))
print(rmse)
```

```
Slope =  0.0006080841195374458
Intercept =  0.0006080841195374458
Enter a random population to get predicted Violent Crime Per Pop
20000
When the population =  20000  Violent Crime Per Pop =  570.8338248557552
7668.12010895052
```

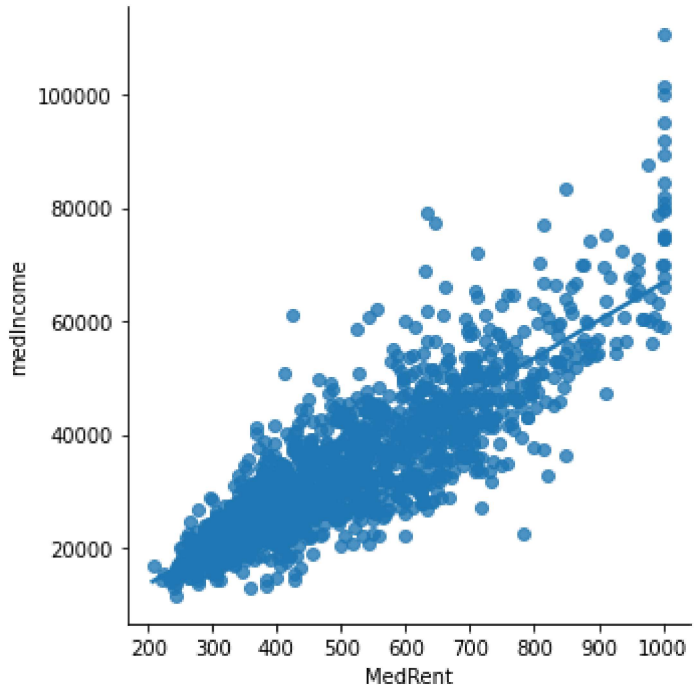
```
from google.colab import drive
drive.mount('/content/drive')
```

```
from scipy import stats
slope, intercept, r, p, std_err = stats.linregress(train.MedRent,train.medIncome)
print('Slope = ',slope)
print('Intercept = ',slope)
def myfunc(x):
  return slope * x + intercept

print("Enter a random medRent to get predicted medIncome")
pop=int(input())
crime = myfunc(pop)
print("When the medRent = ",pop," medIncome = ", crime)

import math
from sklearn.metrics import mean_squared_error
rmse = math.sqrt(mean_squared_error([670,35545],[pop,crime]))
print(rmse)
```

```
Slope =  66.50348186436204
Intercept =  66.50348186436204
Enter a random medRent to get predicted medIncome
2000
When the medRent =  2000  medIncome =  133274.1225228662
69111.32428585563
```

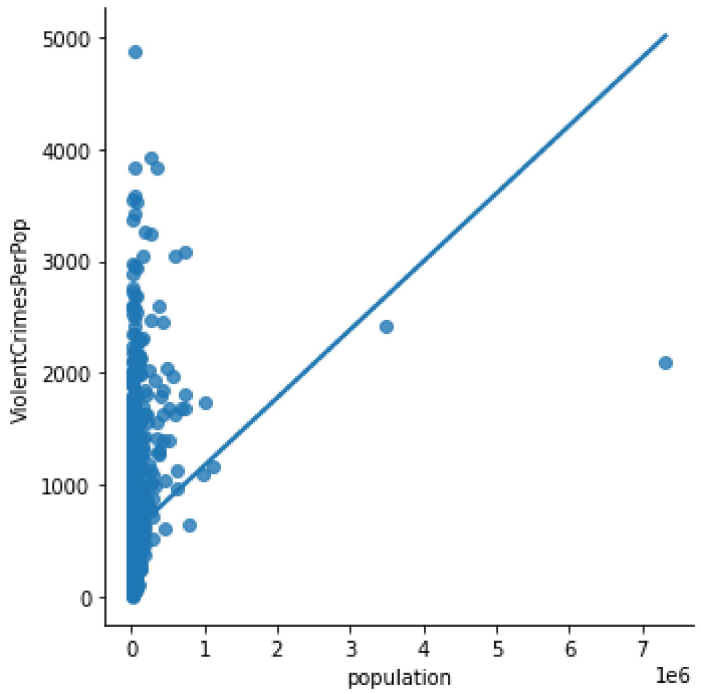
```
import seaborn as sb

# load data
df = train

# use lmplot
sb.lmplot(x = "MedRent",
            y = "medIncome",
            ci = None,
            data = df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f39c4f8b490>
```



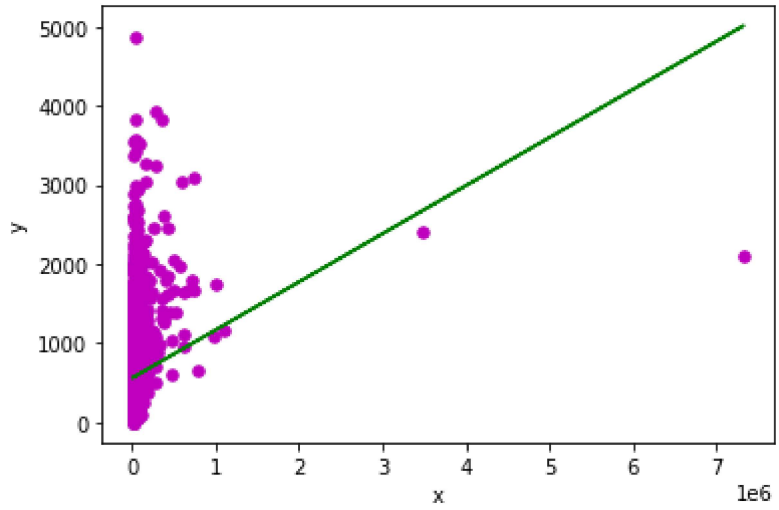

```
import seaborn as sb

# load data
df = train

# use lmplot
sb.lmplot(x = "population",
            y = "ViolentCrimesPerPop",
            ci = None,
            data = df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f39c441e210>
```

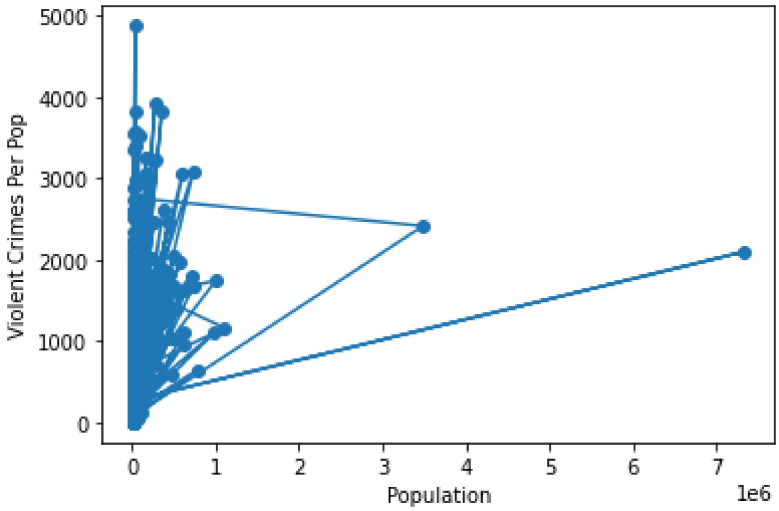



```
def estimate_coef(x, y):
    n = np.size(x)
    m_x = np.mean(x)
    m_y = np.mean(y)
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x

    return (b_0, b_1)

def plot_regression_line(x, y, b):
    plt.scatter(x, y, color = "m",
                marker = "o", s = 30)

    y_pred = b[0] + b[1]*x

    plt.plot(x, y_pred, color = "g")

    plt.xlabel('x')
    plt.ylabel('y')
    plt.show()

x=train.population
y=train.ViolentCrimesPerPop
b = estimate_coef(x,y)
print("Estimated coefficients:\nb_0 = {}  \
          \nb_1 = {}".format(b[0], b[1]))
plot_regression_line(x, y, b)
```

```
Estimated coefficients:
b_0 = 558.6721424650065
b_1 = 0.0006080841195374455
```




```
plt.scatter(train.population,train.ViolentCrimesPerPop)
plt.plot(train.population,train.ViolentCrimesPerPop)
plt.xlabel('Population')
plt.ylabel('Violent Crimes Per Pop')
plt.show()
```




```
import math

from sklearn.metrics import mean_squared_error

rmse = math.sqrt(mean_squared_error([100],[200]))
print(rmse)
```

```
100.0
```

```
from scipy import stats
slope, intercept, r, p, std_err = stats.linregress(train.population,train.ViolentCrimesPerPop
print('Slope = ',slope)
```

```
print('Intercept = ',slope)
def myfunc(x):
  return slope * x + intercept

crime = myfunc(14985)
print("When the population = ",14985," Violent Crime Per Pop = ", crime)
import math
from sklearn.metrics import mean_squared_error
rmse = math.sqrt(mean_squared_error([428.64],[crime]))
print(rmse)
```

```
Slope =  0.0006080841195374458
Intercept =  0.0006080841195374458
When the population =  14985  Violent Crime Per Pop =  567.784282996275
98.3898660700061
```