Import data Using Pandas

```
import pandas as pn
url ='https://raw.githubusercontent.com/swakkhar/'
filename='MachineLearning/master/Codes/linear.csv'
data = pn.read_csv(url+filename,header=None)
```

```
data
```

	0	1
0	0.067732	3.176513
1	0.427810	3.816464
2	0.995731	4.550095
3	0.738336	4.256571
4	0.981083	4.560815
...	...	...
195	0.257017	3.585821
196	0.833735	4.374394
197	0.070095	3.213817
198	0.527070	3.952681
199	0.116163	3.129283

200 rows × 2 columns

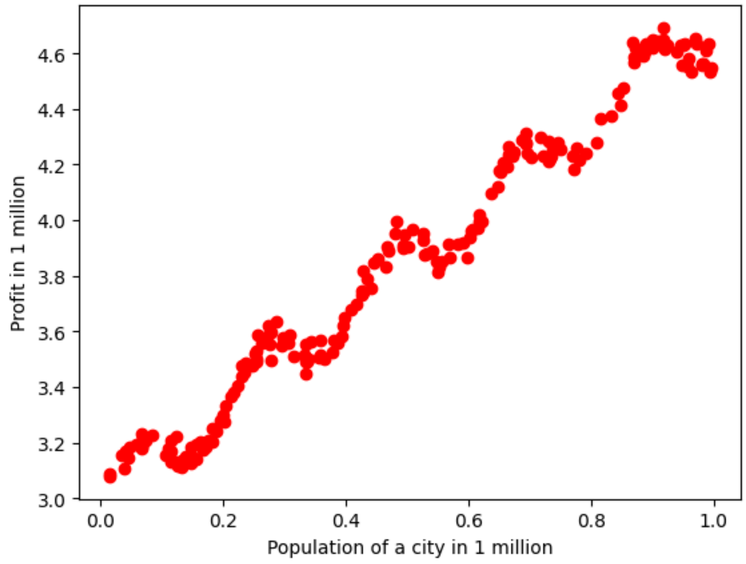
Use Numpy to convert the numerical data

```
import numpy  as np
data=np.asarray(data)
X = np.delete(data, data.shape[1] - 1, axis=1)
y = data[:, -1]
print("X.shape:",X.shape)
print("y.shape:",y.shape)
```

```
X.shape: (200, 1)
y.shape: (200,)
```

Visualize Data

```
import matplotlib
import matplotlib.pyplot as plt

plt.scatter(X,y,color='r')
plt.xlabel("Population of a city in 1 million")
plt.ylabel("Profit in 1 million")
plt.show()
```

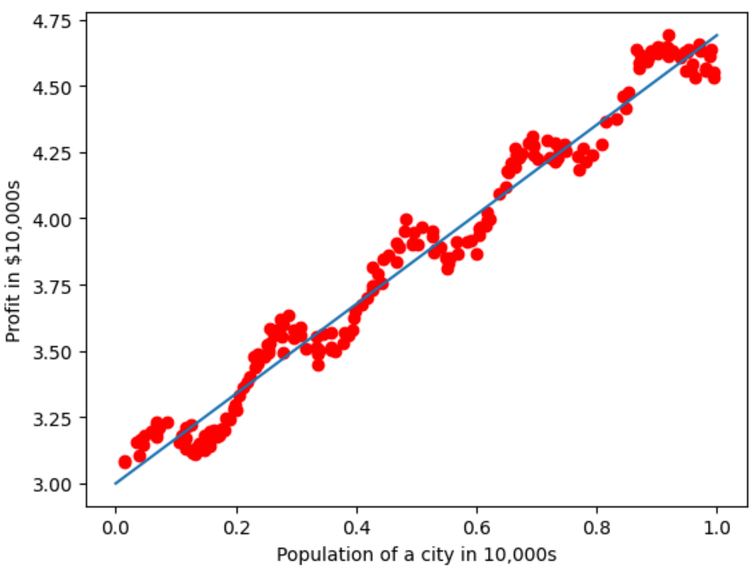



```
for i in range(X.shape[0]):
  print(X[i,0],y[i],'a')
```

```
0.067732 3.176513 a
0.42781 3.816464 a
0.995731 4.550095 a
0.738336 4.256571 a
0.981083 4.560815 a
0.526171 3.929515 a
0.378887 3.52617 a
0.033859 3.156393 a
0.132791 3.110301 a
0.138306 3.149813 a
0.247809 3.476346 a
0.64827 4.119688 a
0.731209 4.282233 a
0.236833 3.486582 a
0.969788 4.655492 a
0.607492 3.965162 a
0.358622 3.5149 a
0.147846 3.125947 a
0.63782 4.094115 a
0.230372 3.476039 a
0.070237 3.21061 a
0.067154 3.190612 a
0.925577 4.631504 a
0.717733 4.29589 a
0.015371 3.085028 a
0.33507 3.44808 a
0.040486 3.16744 a
0.212575 3.364266 a
0.617218 3.993482 a
0.541196 3.891471 a
0.045353 3.143259 a
0.126762 3.114204 a
0.556486 3.851484 a
0.901144 4.621899 a
0.958476 4.580768 a
0.274561 3.620992 a
0.394396 3.580501 a
0.87248 4.618706 a
0.409932 3.676867 a
0.908969 4.641845 a
0.166819 3.175939 a
0.665016 4.26498 a
0.263727 3.558448 a
0.231214 3.436632 a
0.552928 3.831052 a
0.047744 3.182853 a
0.365746 3.498906 a
0.495002 3.946833 a
0.493466 3.900583 a
0.792101 4.238522 a
0.76966 4.23308 a
0.251821 3.521557 a
0.181951 3.203344 a
0.808177 4.278105 a
0.334116 3.555705 a
0.33863 3.502661 a
0.452584 3.859776 a
0.69477 4.275956 a
```

Try a random line

```
w0=3
w1= 1.69
population=np.linspace(0,1,100)
profit=w0+w1*population
import matplotlib
import matplotlib.pyplot as plt

plt.scatter(X,y,color='r')
plt.xlabel("Population of a city in 10,000s")
plt.ylabel("Profit in $10,000s")
plt.plot(population,profit)



#w0=3
#w1= 1
#profit=w0+w1*population
#plt.plot(population,profit)

#w0=3
#w1= 2
#profit=w0+w1*population
#plt.plot(population,profit)

#plt.legend(['dataset','y=3+1.69x','y=3+x','y=3+2x'])
plt.show()
```

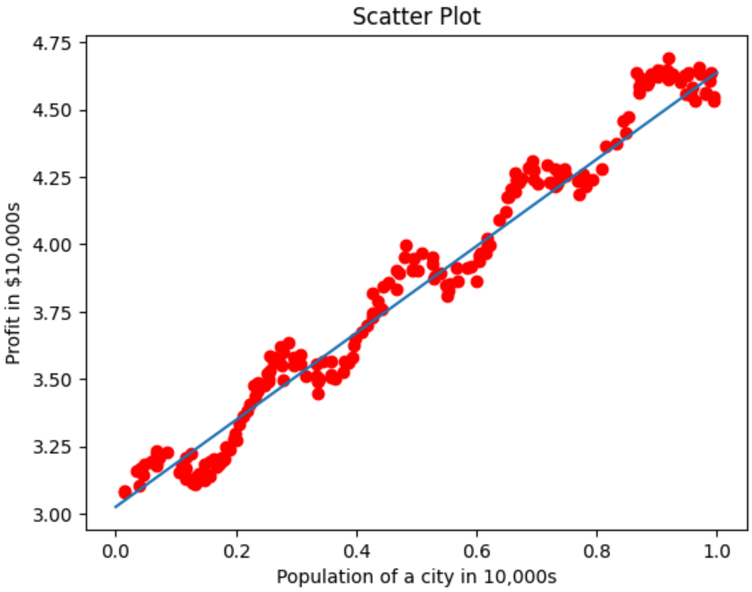


Learn the line using Gradient Descent

```
from numpy import mat
import copy

def learn(X,y,alpha,maxIter):

  # make a deep copy of everything
  X=copy.deepcopy(X)
  y=copy.deepcopy(y)

  # add ones to each of the instances
  ones=np.ones((X.shape[0],1))
  X=np.concatenate((ones,X),axis=1)
  X=mat(X)
  y=mat(y)
  # initialize wights
  w=np.zeros((X.shape[1],1))#np.random.rand(1,X.shape[1])*-1

  w=mat(w)
  for i in range(0,maxIter):
    # for each of the training examples find prediction
```

```
    ypred = X*w
    # for each of the training examples find error
    delY=ypred-y.T
    # take the gradient
    delw=X.T * delY
    # adjust weights using the gradients
    w=w-delw*alpha #/X.shape[0]
  return w

w=learn(X,y,0.000001,20000)
print("w0=",w[0],"w1=",w[1])


population=np.linspace(0,1,1000)
profit=w[0,0]+population*w[1,0]
import matplotlib
import matplotlib.pyplot as plt

plt.scatter(X,y,color='r')
plt.xlabel("Population of a city in 10,000s")
plt.title("Scatter Plot")
plt.ylabel("Profit in $10,000s")
plt.plot(population,profit)
plt.show()
```

```
w0= [[3.02504402]] w1= [[1.61338927]]
```




```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X, y)
print("w0=",reg.intercept_)
print("w1=",reg.coef_)
```

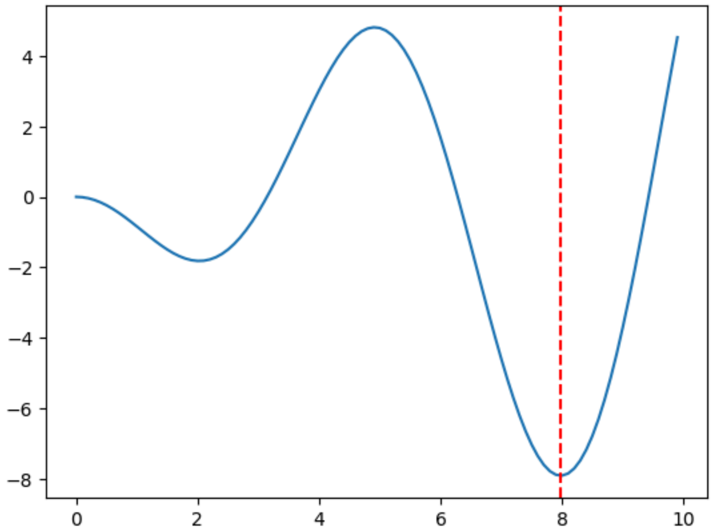
```
w0= 3.007743242697591
w1= [1.69532264]
```

```
reg.predict(np.array([[0.5]]))
```

```
array([3.85540456])
```

```
# multimodal function
from numpy import sin
from numpy import arange
from matplotlib import pyplot

# objective function
def objective(x):
    return -x * sin(x)

# define range for input
r_min, r_max = 0.0, 10.0
# sample input range uniformly at 0.1 increments
inputs = arange(r_min, r_max, 0.1)
# compute targets
results = objective(inputs)
# create a line plot of input vs result
pyplot.plot(inputs, results)
```

```
# define optimal input value
x_optima = 7.9787
# draw a vertical line at the optimal input
pyplot.axvline(x=x_optima, ls='--', color='red')
# show the plot
pyplot.show()
```



```
inputs
```

```
array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. , 1.1, 1.2,
       1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2. , 2.1, 2.2, 2.3, 2.4, 2.5,
       2.6, 2.7, 2.8, 2.9, 3. , 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8,
       3.9, 4. , 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5. , 5.1,
       5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 6. , 6.1, 6.2, 6.3, 6.4,
       6.5, 6.6, 6.7, 6.8, 6.9, 7. , 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7,
       7.8, 7.9, 8. , 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 9. ,
       9.1, 9.2, 9.3, 9.4, 9.5, 9.6, 9.7, 9.8, 9.9])
```

```
results
```

```
array([-0.        , -0.00998334, -0.03973387, -0.08865606, -0.15576734,
       -0.23971277, -0.33878548, -0.45095238, -0.57388487, -0.70499422,
       -0.84147098, -0.9803281 , -1.1184469 , -1.25262564, -1.37962962,
       -1.49624248, -1.59931776, -1.68583018, -1.75292574, -1.79797017,
       -1.81859485, -1.81273967, -1.77869209, -1.71512199, -1.62111163,
       -1.49618036, -1.34030357, -1.15392568, -0.93796682, -0.69382305,
       -0.42336002, -0.12890005,  0.18679726,  0.52056079,  0.86883975,
        1.2277413 ,  1.5930736 ,  1.96039372,  2.32505999,  2.68228802,
        3.02720998,  3.35493616,  3.66061824,  3.93951353,  4.18704913,
        4.39888553,  4.57097862,  4.69963931,  4.78159012,  4.8140178 ,
        4.79462137,  4.72165488,  4.59396421,  4.41101744,  4.17292823,
        3.88047179,  3.53509317,  3.13890759,  2.69469264,  2.20587232,
        1.67649299,  1.11119128,  0.5151543 , -0.10592757, -0.74591491,
       -1.39827992, -2.056173  , -2.71249447, -3.35997079, -3.99123437,
       -4.59890619, -5.17568018, -5.71440862, -6.20818733, -6.65043991,
       -7.03499983, -7.35618951, -7.6088954 , -7.78863809, -7.8916366 ,
       -7.91486597, -7.85610747, -7.71399056, -7.48802622, -7.17863083,
       -6.78714046, -6.31581504, -5.7678323 , -5.1472713 , -4.45908562,
       -3.70906637, -2.9037951 , -2.05058721, -1.15742614, -0.232889  ,
        0.71393564,  1.6735371 ,  2.63607808,  3.59149547,  4.52960535])
```

```
r=arange(2.5,4.6 , 0.2)
```

```
v=objective(r)
```

```
for i in range(len(r)):
  print(r[i],v[i],"c")
```

```
2.5 -1.4961803602598913 c
2.7 -1.1539256766313404 c
2.9000000000000004 -0.6938230547205478 c
3.1000000000000005 -0.12890005354319917 c
3.3000000000000007 0.5205607906727221 c
3.500000000000001 1.2277412969136727 c
3.700000000000001 1.9603937213614286 c
3.9000000000000012 2.6822880208175026 c
4.100000000000001 3.3549361553640877 c
4.300000000000002 3.939513528022661 c
4.500000000000002 4.39888552949294 c
```

Start coding or generate with AI.