

Name: Mallikarjun Katakam  
Roll no.: 2018IMT-046  
Course: Machine Learning  
Course Code: ITIT 4107-2021  
Assignment-4  
Deadline: 18th Oct 2021

## Problem:

Suppose you are the CEO of a restaurant franchise and are considering different cities for opening a new outlet. The chain already has trucks in various cities and you have data for profits and populations from the cities. You would like to use this data to help you select which city to expand to next. The file `ex1data1.txt` contains the dataset for our linear regression problem. The first column is the population of a city and the second column is the profit of a food truck in that city. A negative value for profit indicates a loss.

1. Use a scatter plot to visualize the data, since it has only two properties to plot (profit and population).
2. Consider a simple linear model with two parameters and one input variable and mean square error cost function to implement the gradient descent algorithm to find the intercepts. Assume a suitable terminating condition.
3. Plot the model alongside the scatterplot to show the fit model.
4. Perform steps 1,2,3 in batch mode for varying values of alpha, learning rate and plot the results.
5. For each of the experiments performed above in steps 1,2,3,4 with varying learning rates visualize the cost function as a contour plot as well as plot the values of parameters to visualize the stepwise traversal of the parameters on this contour plot.

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
In [2]: df=pd.read_csv("https://raw.githubusercontent.com/AlfTang/Linear-Regression/master/ex1data1.txt",names = ['population','profit'],header = None)
```

```
In [3]: df.head()
```

Out[3]:

	population	profit
0	6.1101	17.5920
1	5.5277	9.1302
2	8.5186	13.6620
3	7.0032	11.8540
4	5.8598	6.8233

```
In [4]: df.describe()
```

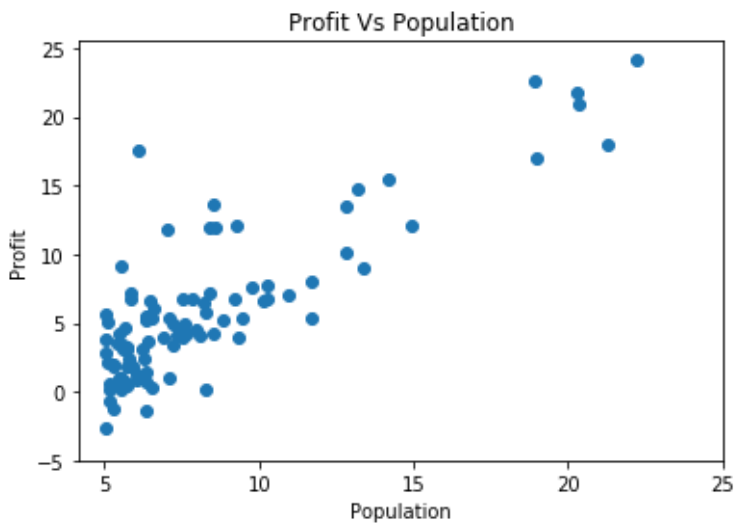
```
Out[4]:
```

	population	profit
count	97.000000	97.000000
mean	8.159800	5.839135
std	3.869884	5.510262
min	5.026900	-2.680700
25%	5.707700	1.986900
50%	6.589400	4.562300
75%	8.578100	7.046700
max	22.203000	24.147000

## Visualizing the Data

```
In [5]: plt.scatter(df['population'],df['profit'])  
plt.title("Profit Vs Population")  
plt.xticks(np.arange(5,30,step=5))  
plt.yticks(np.arange(-5,30,step=5))  
plt.xlabel("Population")  
plt.ylabel("Profit")
```

```
Out[5]: Text(0,0.5,'Profit')
```



```
In [6]: def computeCost(X,y,theta):  
        m=len(y)  
        predictions=X.dot(theta)  
        square_err=(predictions - y)**2  
  
        return 1/(2*m) * np.sum(square_err)
```

```
In [7]: data_n=df.values
m=len(data_n[:, -1])
X=np.append(np.ones((m,1)),data_n[:,0].reshape(m,1),axis=1)
y=data_n[:,1].reshape(m,1)
theta=np.zeros((2,1))
computeCost(X,y,theta)
```

Out[7]: 32.072733877455676

```
In [8]: def gradientDescent(X,y,theta,alpha,num_iters):

    m=len(y)
    J_history=[]

    for i in range(num_iters):
        predictions = X.dot(theta)
        error = np.dot(X.transpose(),(predictions - y))
        descent=alpha * 1/m * error
        theta-=descent
        J_history.append(computeCost(X,y,theta))

    return theta, J_history
```

```
In [11]: def plot_model_fit(alpha,theta):
    plt.scatter(df['population'],df['profit'])
    plt.title("Profit vs Population with alpha = {}".format(alpha))
    x_value=[x for x in range(25)]
    y_value=[y*theta[1]+theta[0] for y in x_value]
    plt.plot(x_value,y_value,color="r")
    plt.xticks(np.arange(5,30,step=5))
    plt.yticks(np.arange(-5,30,step=5))
    plt.xlabel("Population")
    plt.ylabel("Profit")

def func(i):
    theta=np.zeros((2,1))
    alpha = alpha_values[i]
    theta,J_history = gradientDescent(X,y,theta,alpha,100)
    print('The ALPHA value, learning_rate: {}'.format(alpha))
    print('The THETA value: {}'.format(theta))
    print('\nHypothesis Function:')
    print("h(x) =" +str(round(theta[0,0],2))+" + " +str(round(theta[1,0],2))+"x1")
    J_histories.append(J_history)
    plot_model_fit(alpha,theta)
```

```
In [12]: alpha_values = np.arange(0.001, 0.02, 0.002)
J_histories = []

print("Alpha values: ({}).format(len(alpha_values)), alpha_values)
func(0)
```

Alpha values: (10) [0.001 0.003 0.005 0.007 0.009 0.011 0.013 0.015 0.017 0.019]

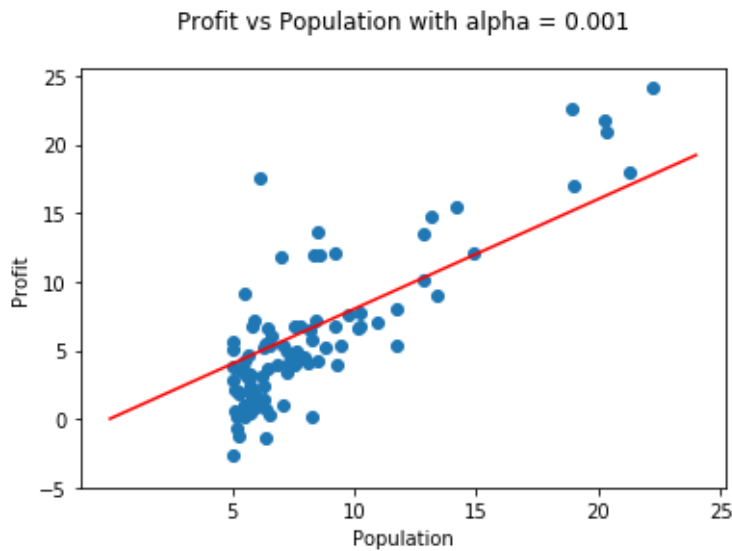
The ALPHA value, learning\_rate: 0.001

The THETA value: [[0.00868909]

[0.80063674]]

Hypothesis Function:

$h(x) = 0.01 + 0.8x_1$



```
In [13]: func(1)
```

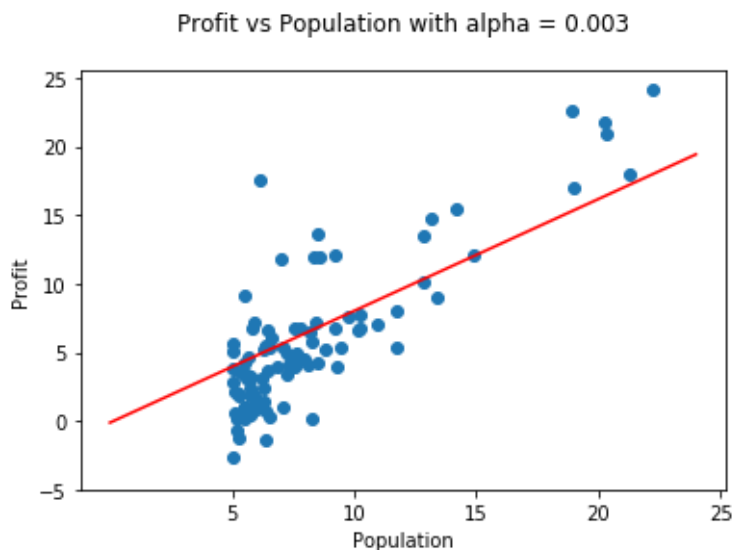
The ALPHA value, learning\_rate: 0.003

The THETA value: [[-0.12960188]

[ 0.81468011]]

Hypothesis Function:

$h(x) = -0.13 + 0.81x_1$



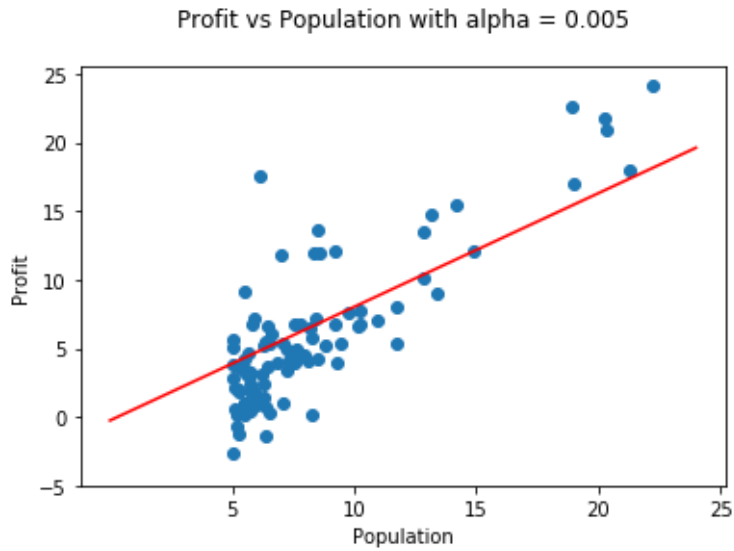
```
In [14]: func(2)
```

The ALPHA value, learning\_rate: 0.005

The THETA value:  $\begin{bmatrix} -0.26305598 \\ 0.82808702 \end{bmatrix}$

Hypothesis Function:

$h(x) = -0.26 + 0.83x_1$



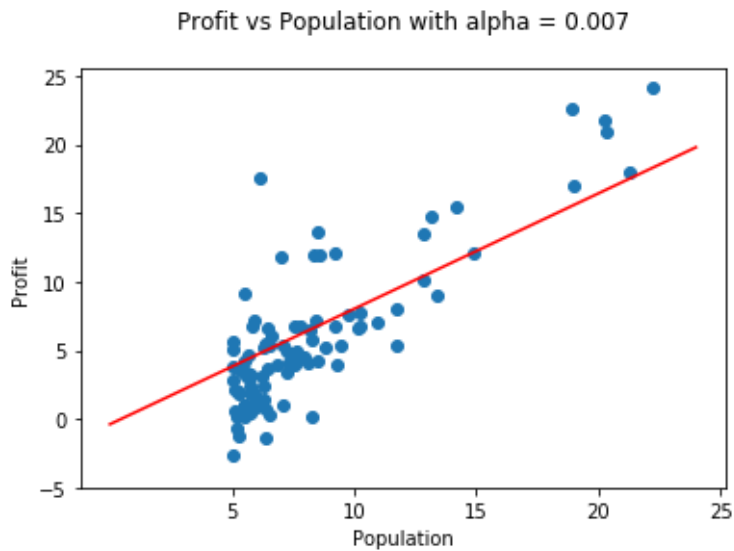
```
In [15]: func(3)
```

The ALPHA value, learning\_rate: 0.007

The THETA value:  $\begin{bmatrix} -0.39182677 \\ 0.84102345 \end{bmatrix}$

Hypothesis Function:

$h(x) = -0.39 + 0.84x_1$



```
In [16]: func(4)
```

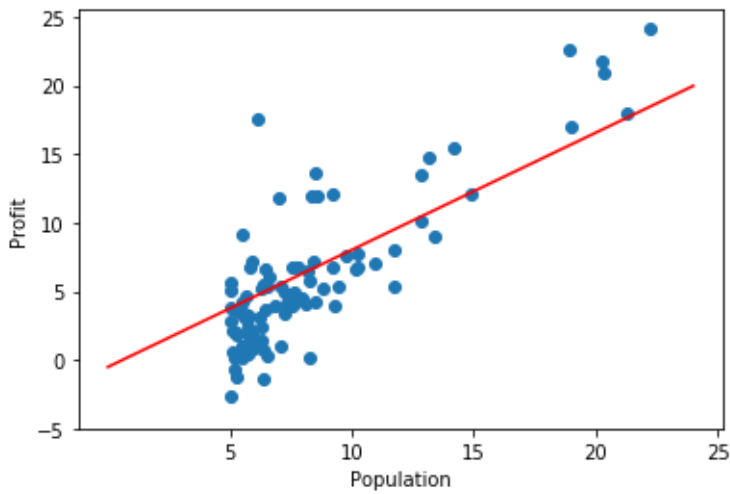
The ALPHA value, learning\_rate: 0.009000000000000001

The THETA value:  $\begin{bmatrix} -0.516077 \\ 0.85350573 \end{bmatrix}$

Hypothesis Function:

$h(x) = -0.52 + 0.85x_1$

Profit vs Population with alpha = 0.009000000000000001



```
In [17]: func(5)
```

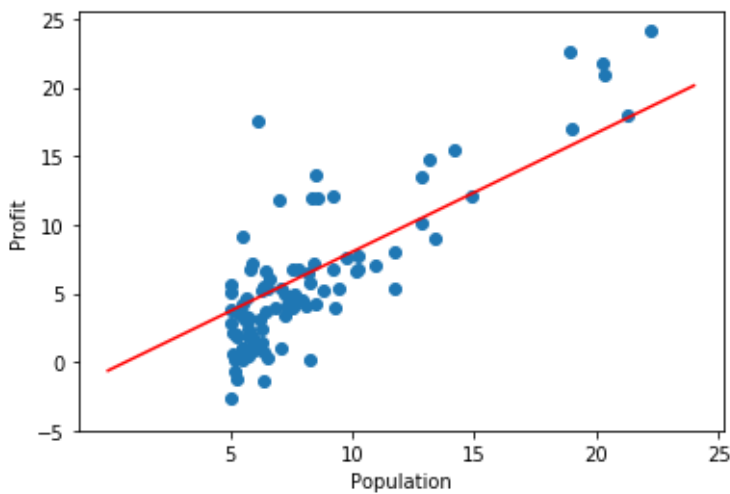
The ALPHA value, learning\_rate: 0.011

The THETA value:  $\begin{bmatrix} -0.63596383 \\ 0.86554966 \end{bmatrix}$

Hypothesis Function:

$h(x) = -0.64 + 0.87x_1$

Profit vs Population with alpha = 0.011



In [18]: func(6)

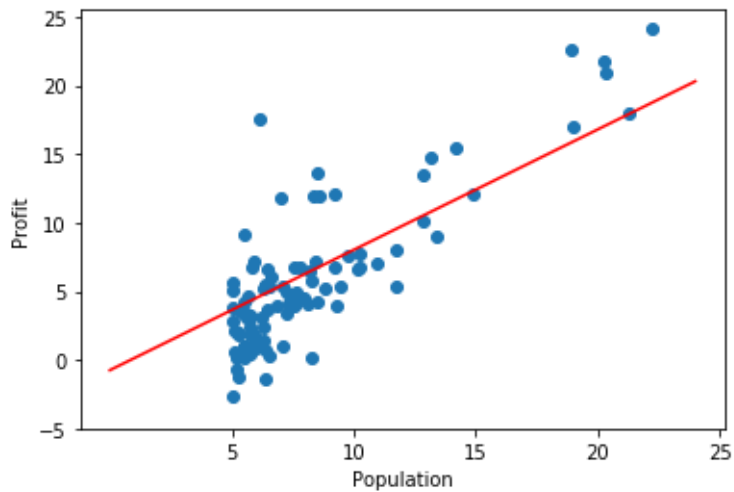
The ALPHA value, learning\_rate: 0.013000000000000001

The THETA value:  $\begin{bmatrix} -0.75163899 \\ 0.87717049 \end{bmatrix}$

Hypothesis Function:

$$h(x) = -0.75 + 0.88x_1$$

Profit vs Population with alpha = 0.013000000000000001



In [19]: func(7)

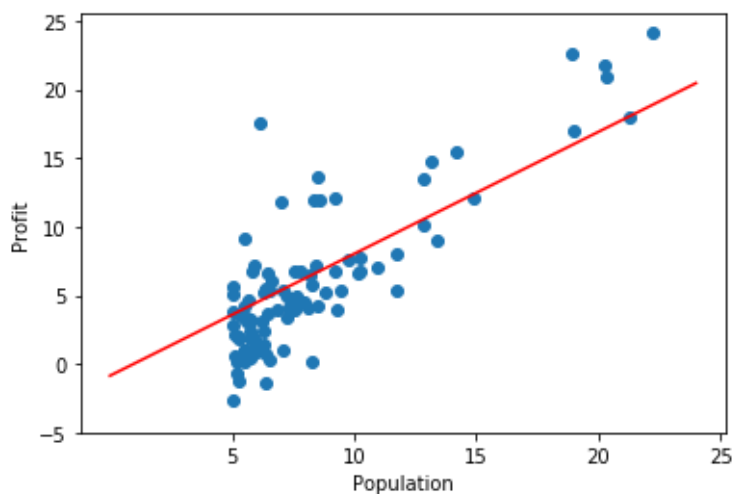
The ALPHA value, learning\_rate: 0.015

The THETA value:  $\begin{bmatrix} -0.863249 \\ 0.88838292 \end{bmatrix}$

Hypothesis Function:

$$h(x) = -0.86 + 0.89x_1$$

Profit vs Population with alpha = 0.015



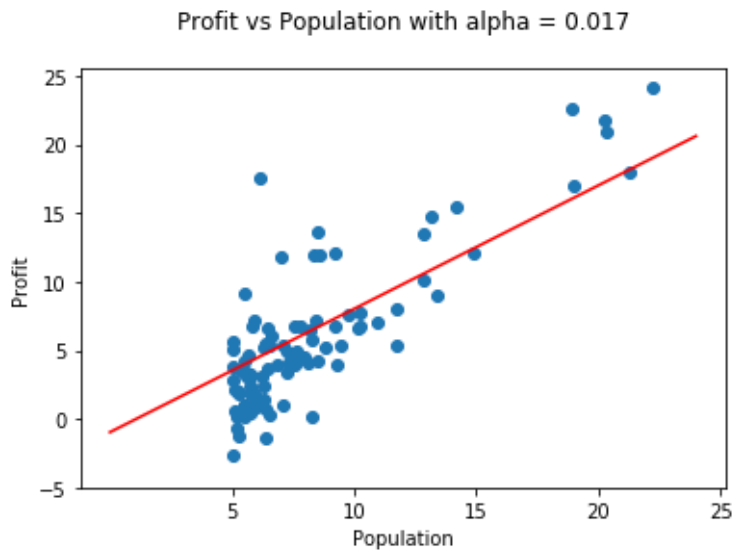
In [20]: `func(8)`

The ALPHA value, learning\_rate: 0.017

The THETA value:  $\begin{bmatrix} -0.97093532 \\ 0.89920118 \end{bmatrix}$

Hypothesis Function:

$h(x) = -0.97 + 0.9x$



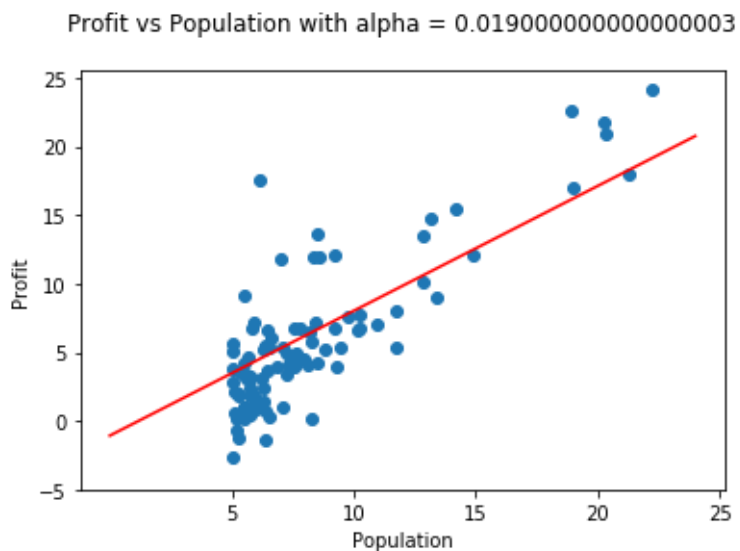
In [21]: `func(9)`

The ALPHA value, learning\_rate: 0.019000000000000003

The THETA value:  $\begin{bmatrix} -1.07483455 \\ 0.90963899 \end{bmatrix}$

Hypothesis Function:

$h(x) = -1.07 + 0.91x$





```
In [22]: x = np.arange(1,125,1.25)
y = alpha_values
len(x)
```

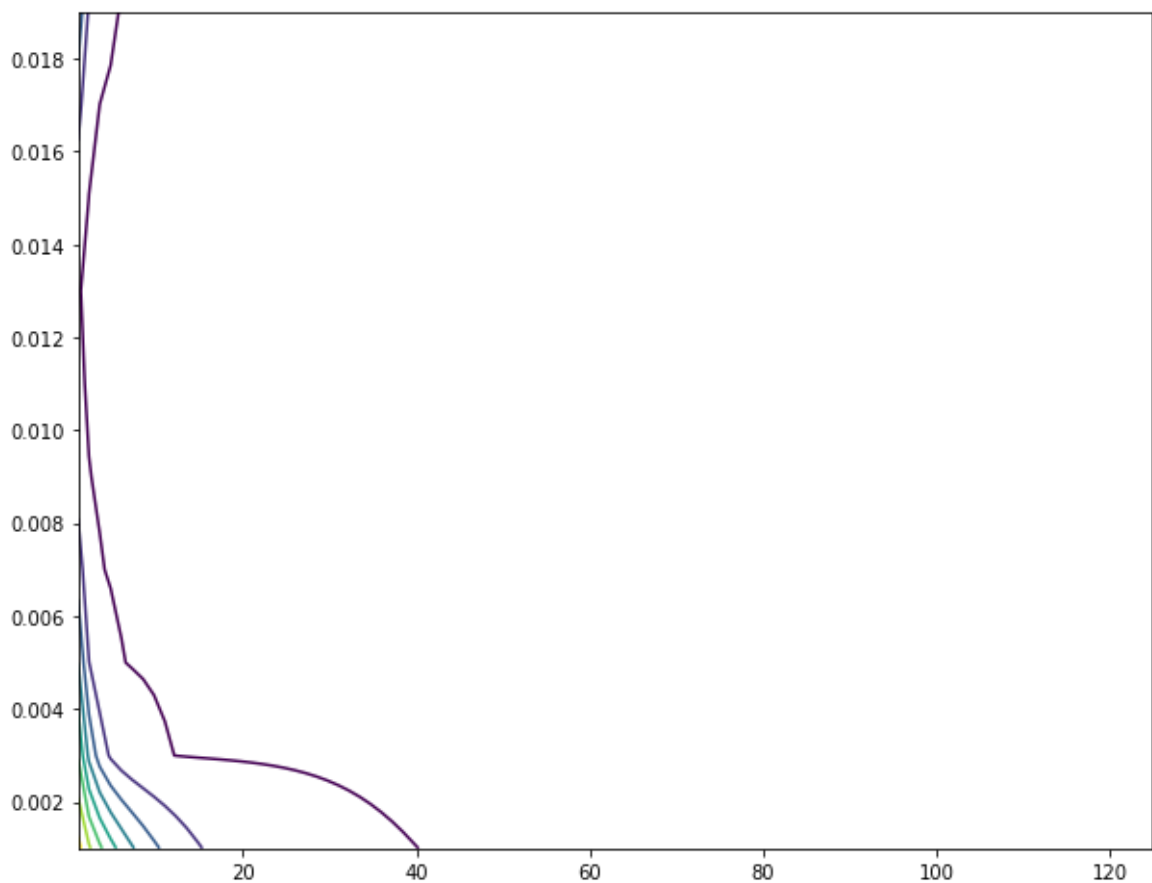
Out[22]: 100

```
In [23]: print(min(min(J_histories)))
max(max(J_histories))
```

5.376953190466999

Out[23]: 27.94761974682486

```
In [24]: z = J_histories
fig, ax = plt.subplots(1, 1, figsize=(10, 8))
ax.contour(x,y,z)
plt.show()
```

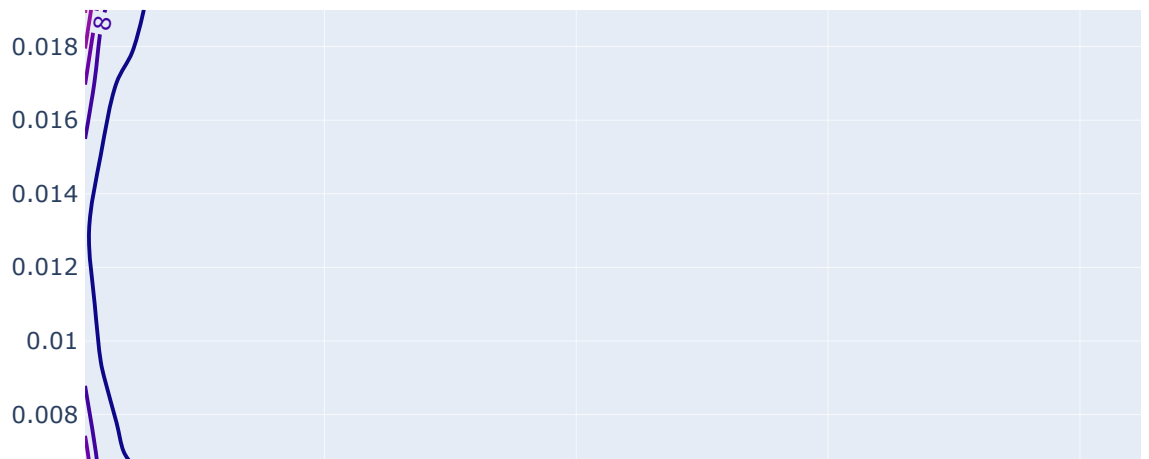


```
In [25]: import plotly.graph_objects as go
import plotly.offline as pyo
pyo.init_notebook_mode()

fig = go.Figure(data=go.Contour(
    x=x, y=y, z=z,
    contours=dict(
        coloring='lines',
        showlabels=True),
    line_width=2
))

fig.show()

# x -> Value of the cost function
# y -> Learning rates
# z -> The value of the cost function at a given Learning rate alpha
```



## Inference

If slope is +ve :  $\theta_j = \theta_j - (+ve \text{ value})$ . Hence value of  $\theta_j$  decreases.

If slope is -ve :  $\theta_j = \theta_j - (-ve \text{ value})$ . Hence value of  $\theta_j$  increases.

The choice of correct learning rate is very important as it ensures that Gradient Descent converges in a reasonable time:

- If we choose  $\alpha$  to be very large, Gradient Descent can overshoot the minimum. It may fail to converge or even diverge.
- If we choose  $\alpha$  to be very small, Gradient Descent will take small steps to reach local minima and will take a longer time to reach minima.

Code Link: <https://github.com/Mallik813/ITIT-4103-2021/tree/main/Assignment-4> (<https://github.com/Mallik813/ITIT-4103-2021/tree/main/Assignment-4>)

In [ ]: