HW0

September 18, 2021

[60]:
```
print('NAME: SOMTO A. \nID:801215537 \nHW#: 0\nGITHUB: https://github.com/
 ↪Somto-Dera/ECGR5090-Machine-Learning ')


# Data Visualisation
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import seaborn as sns
import scipy.stats as stats
from scipy import stats

from matplotlib import pyplot as plt

from sklearn.preprocessing import MinMaxScaler


# Ensure plots are displayed inline in the notebook
%matplotlib inline

from matplotlib import pyplot as plt
```

```
NAME: SOMTO A.
ID:801215537
HW#: 0
GITHUB: https://github.com/Somto-Dera/ECGR5090-Machine-Learning
```

[61]:
```
#housing = pd.DataFrame(pd.read_csv("Housing.csv"))
#housing.head()
#Load the dataset from my workspace
hw1 = pd.read_csv('D3.csv')
#print(df)

#Display dataset
df.head()
hw1
```

```
#df.head() # To get first n rows from the dataset default value of n is 5
#M=len(df)
#M
```

[61]:

	x1	x2	x3	y
0	0.000000	3.440000	0.440000	4.387545
1	0.040404	0.134949	0.888485	2.679650
2	0.080808	0.829899	1.336970	2.968490
3	0.121212	1.524848	1.785455	3.254065
4	0.161616	2.219798	2.233939	3.536375
..	…	…	…	…
95	3.838384	1.460202	3.046061	-4.440595
96	3.878788	2.155152	3.494545	-4.458663
97	3.919192	2.850101	3.943030	-4.479995
98	3.959596	3.545051	0.391515	-3.304593
99	4.000000	0.240000	0.840000	-5.332455

[100 rows x 4 columns]

[62]:
```
x1 = df.values[:, 0]  # get input values from first column
x2 = df.values[:, 1]  # get input values from second column
x3 = df.values[:, 2]  # get input values from third column
y = df.values[:, 3]   # get output values from fourth column
m = len(y) # Number of training examples
print('X1 = ', x1[:]) # Show only first 5 records
print('X2 = ', x2[:]) # Show only first 5 records
print('X3 = ', x2[:]) # Show only first 5 records
print('y = ', y[:])
print('m = ', m)
```

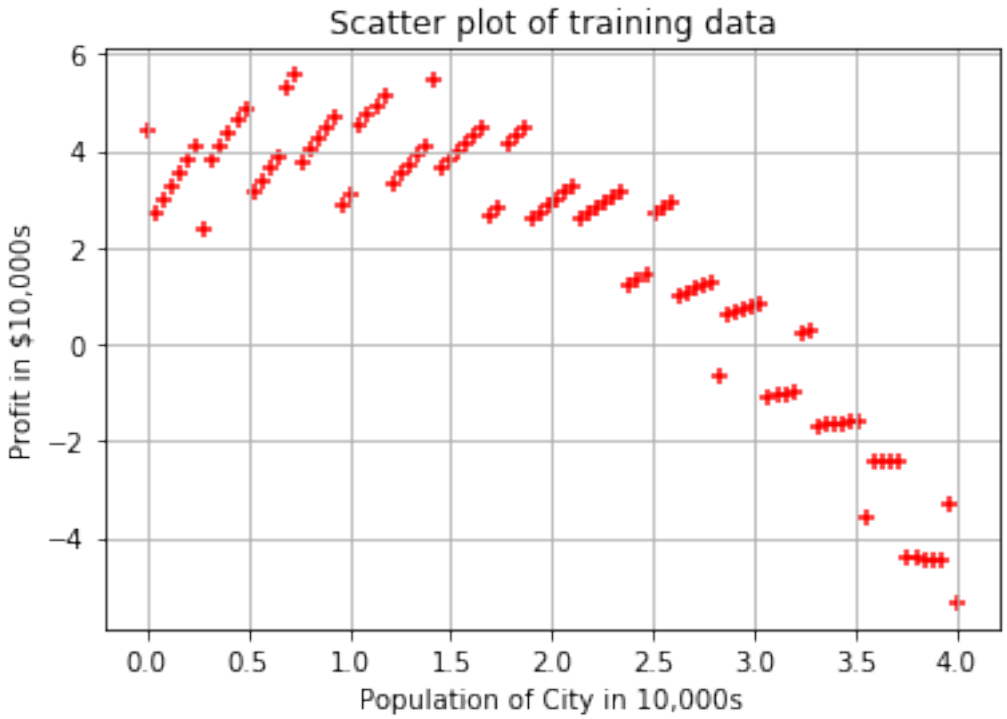
```
X1 =  [0.         0.04040404 0.08080808 0.12121212 0.16161616 0.2020202
 0.24242424 0.28282828 0.32323232 0.36363636 0.4040404  0.44444444
 0.48484848 0.52525253 0.56565657 0.60606061 0.64646465 0.68686869
 0.72727273 0.76767677 0.80808081 0.84848485 0.88888889 0.92929293
 0.96969697 1.01010101 1.05050505 1.09090909 1.13131313 1.17171717
 1.21212121 1.25252525 1.29292929 1.33333333 1.37373737 1.41414141
 1.45454545 1.49494949 1.53535354 1.57575758 1.61616162 1.65656566
 1.6969697  1.73737374 1.77777778 1.81818182 1.85858586 1.8989899
 1.93939394 1.97979798 2.02020202 2.06060606 2.1010101  2.14141414
 2.18181818 2.22222222 2.26262626 2.3030303  2.34343434 2.38383838
 2.42424242 2.46464646 2.50505051 2.54545455 2.58585859 2.62626263
 2.66666667 2.70707071 2.74747475 2.78787879 2.82828283 2.86868687
 2.90909091 2.94949495 2.98989899 3.03030303 3.07070707 3.11111111
 3.15151515 3.19191919 3.23232323 3.27272727 3.31313131 3.35353535
 3.39393939 3.43434343 3.47474747 3.51515152 3.55555556 3.5959596
 3.63636364 3.67676768 3.71717172 3.75757576 3.7979798  3.83838384
 3.87878788 3.91919192 3.95959596 4.        ]
```

```
X2 =  [3.44       0.13494949 0.82989899 1.52484848 2.21979798 2.91474747
 3.60969697 0.30464646 0.99959596 1.69454545 2.38949495 3.08444444
 3.77939394 0.47434343 1.16929293 1.86424242 2.55919192 3.25414141
 3.94909091 0.6440404  1.3389899  2.03393939 2.72888889 3.42383838
 0.11878788 0.81373737 1.50868687 2.20363636 2.89858586 3.59353535
 0.28848485 0.98343434 1.67838384 2.37333333 3.06828283 3.76323232
 0.45818182 1.15313131 1.84808081 2.5430303  3.2379798  3.93292929
 0.62787879 1.32282828 2.01777778 2.71272727 3.40767677 0.10262626
 0.79757576 1.49252525 2.18747475 2.88242424 3.57737374 0.27232323
 0.96727273 1.66222222 2.35717172 3.05212121 3.74707071 0.4420202
 1.1369697  1.83191919 2.52686869 3.22181818 3.91676768 0.61171717
 1.30666667 2.00161616 2.69656566 3.39151515 0.08646465 0.78141414
 1.47636364 2.17131313 2.86626263 3.56121212 0.25616162 0.95111111
 1.64606061 2.3410101  3.0359596  3.73090909 0.42585859 1.12080808
 1.81575758 2.51070707 3.20565657 3.90060606 0.59555556 1.29050505
 1.98545455 2.68040404 3.37535354 0.07030303 0.76525253 1.46020202
 2.15515152 2.85010101 3.54505051 0.24      ]
X3 =  [3.44       0.13494949 0.82989899 1.52484848 2.21979798 2.91474747
 3.60969697 0.30464646 0.99959596 1.69454545 2.38949495 3.08444444
 3.77939394 0.47434343 1.16929293 1.86424242 2.55919192 3.25414141
 3.94909091 0.6440404  1.3389899  2.03393939 2.72888889 3.42383838
 0.11878788 0.81373737 1.50868687 2.20363636 2.89858586 3.59353535
 0.28848485 0.98343434 1.67838384 2.37333333 3.06828283 3.76323232
 0.45818182 1.15313131 1.84808081 2.5430303  3.2379798  3.93292929
 0.62787879 1.32282828 2.01777778 2.71272727 3.40767677 0.10262626
 0.79757576 1.49252525 2.18747475 2.88242424 3.57737374 0.27232323
 0.96727273 1.66222222 2.35717172 3.05212121 3.74707071 0.4420202
 1.1369697  1.83191919 2.52686869 3.22181818 3.91676768 0.61171717
 1.30666667 2.00161616 2.69656566 3.39151515 0.08646465 0.78141414
 1.47636364 2.17131313 2.86626263 3.56121212 0.25616162 0.95111111
 1.64606061 2.3410101  3.0359596  3.73090909 0.42585859 1.12080808
 1.81575758 2.51070707 3.20565657 3.90060606 0.59555556 1.29050505
 1.98545455 2.68040404 3.37535354 0.07030303 0.76525253 1.46020202
 2.15515152 2.85010101 3.54505051 0.24      ]
y =  [ 4.38754501  2.6796499   2.96848981  3.25406475  3.53637472  3.81541972
  4.09119974  2.36371479  3.83296487  4.09894997  4.3616701   4.62112526
  4.87731544  3.13024065  3.37990089  3.62629616  3.86942645  5.30929177
  5.54589212  3.77922749  4.00929789  4.23610332  4.45964378  4.67991926
  2.89692977  3.1106753   4.52115587  4.72837146  4.93232207  5.13300772
  3.33042839  3.52458409  3.71547481  3.90310057  4.08746135  5.46855715
  3.64638799  3.82095385  3.99225473  4.16029065  4.32506159  4.48656756
  2.64480856  2.79978458  4.15149563  4.29994171  4.44512281  2.58703894
  2.7256901   2.86107628  2.9931975   3.12205373  3.247645    2.56997129
  2.68903261  2.80482896  2.91736034  3.02662674  3.13262817  1.23536462
  1.3348361   1.43104261  2.72398415  2.81366071  2.9000723   0.98321892
  1.06310057  1.13971724  1.21306894  1.28315566 -0.65002258  0.6135342
  0.673826    0.73085284  0.7846147   0.83511159 -1.1176565  -1.07368956
 -1.03298759 -0.99555059  0.23862143  0.26952848 -1.70282944 -1.67845234
```

```
 -1.6573402  -1.63949305 -1.62491086 -1.61359365 -3.60554141 -2.40075414
 -2.39923185 -2.40097453 -2.40598218 -4.41425481 -4.4257924  -4.44059497
 -4.45866252 -4.47999504 -3.30459253 -5.33245499]
m =  100
```

[63]:
```
plt.scatter(x1,y, color='red',marker= '+')
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
plt.xlabel('Population of City in 10,000s')
plt.ylabel('Profit in $10,000s')
plt.title('Scatter plot of training data')
```

[63]:
```
Text(0.5, 1.0, 'Scatter plot of training data')
```



[64]:
```
#Lets create a matrix with single column of ones
X_0 = np.ones((m, 1))
X_0[:]
```

[64]:
```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
```

[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],

```
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.]])
```

[65]:
```
X_1 = x1.reshape(m, 1)
X_1[:]
```

[65]:
```
array([[0.        ],
       [0.04040404],
       [0.08080808],
       [0.12121212],
       [0.16161616],
       [0.2020202 ],
       [0.24242424],
       [0.28282828],
       [0.32323232],
       [0.36363636],
       [0.4040404 ],
       [0.44444444],
       [0.48484848],
       [0.52525253],
       [0.56565657],
       [0.60606061],
       [0.64646465],
       [0.68686869],
       [0.72727273],
       [0.76767677],
       [0.80808081],
       [0.84848485],
       [0.88888889],
       [0.92929293],
       [0.96969697],
       [1.01010101],
       [1.05050505],
       [1.09090909],
       [1.13131313],
       [1.17171717],
       [1.21212121],
       [1.25252525],
       [1.29292929],
       [1.33333333],
       [1.37373737],
       [1.41414141],
       [1.45454545],
       [1.49494949],
       [1.53535354],
       [1.57575758],
       [1.61616162],
       [1.65656566],
       [1.6969697 ],
       [1.73737374],
```

```
[1.77777778],
[1.81818182],
[1.85858586],
[1.8989899 ],
[1.93939394],
[1.97979798],
[2.02020202],
[2.06060606],
[2.1010101 ],
[2.14141414],
[2.18181818],
[2.22222222],
[2.26262626],
[2.3030303 ],
[2.34343434],
[2.38383838],
[2.42424242],
[2.46464646],
[2.50505051],
[2.54545455],
[2.58585859],
[2.62626263],
[2.66666667],
[2.70707071],
[2.74747475],
[2.78787879],
[2.82828283],
[2.86868687],
[2.90909091],
[2.94949495],
[2.98989899],
[3.03030303],
[3.07070707],
[3.11111111],
[3.15151515],
[3.19191919],
[3.23232323],
[3.27272727],
[3.31313131],
[3.35353535],
[3.39393939],
[3.43434343],
[3.47474747],
[3.51515152],
[3.55555556],
[3.5959596 ],
[3.63636364],
```

```
       [3.67676768],
       [3.71717172],
       [3.75757576],
       [3.7979798 ],
       [3.83838384],
       [3.87878788],
       [3.91919192],
       [3.95959596],
       [4.        ]])
```

[66]:
```
# Lets use hstack() function from numpy to stack X_0 and X_1 horizontally (i.e. column wise) to make a single 2D array.
# This will be our final X matrix (feature matrix)
X = np.hstack((X_0, X_1))
X[:]
```

[66]:
```
array([[1.        , 0.        ],
       [1.        , 0.04040404],
       [1.        , 0.08080808],
       [1.        , 0.12121212],
       [1.        , 0.16161616],
       [1.        , 0.2020202 ],
       [1.        , 0.24242424],
       [1.        , 0.28282828],
       [1.        , 0.32323232],
       [1.        , 0.36363636],
       [1.        , 0.4040404 ],
       [1.        , 0.44444444],
       [1.        , 0.48484848],
       [1.        , 0.52525253],
       [1.        , 0.56565657],
       [1.        , 0.60606061],
       [1.        , 0.64646465],
       [1.        , 0.68686869],
       [1.        , 0.72727273],
       [1.        , 0.76767677],
       [1.        , 0.80808081],
       [1.        , 0.84848485],
       [1.        , 0.88888889],
       [1.        , 0.92929293],
       [1.        , 0.96969697],
       [1.        , 1.01010101],
       [1.        , 1.05050505],
       [1.        , 1.09090909],
       [1.        , 1.13131313],
       [1.        , 1.17171717],
       [1.        , 1.21212121],
```

```
[1.        , 1.25252525],
[1.        , 1.29292929],
[1.        , 1.33333333],
[1.        , 1.37373737],
[1.        , 1.41414141],
[1.        , 1.45454545],
[1.        , 1.49494949],
[1.        , 1.53535354],
[1.        , 1.57575758],
[1.        , 1.61616162],
[1.        , 1.65656566],
[1.        , 1.6969697 ],
[1.        , 1.73737374],
[1.        , 1.77777778],
[1.        , 1.81818182],
[1.        , 1.85858586],
[1.        , 1.8989899 ],
[1.        , 1.93939394],
[1.        , 1.97979798],
[1.        , 2.02020202],
[1.        , 2.06060606],
[1.        , 2.1010101 ],
[1.        , 2.14141414],
[1.        , 2.18181818],
[1.        , 2.22222222],
[1.        , 2.26262626],
[1.        , 2.3030303 ],
[1.        , 2.34343434],
[1.        , 2.38383838],
[1.        , 2.42424242],
[1.        , 2.46464646],
[1.        , 2.50505051],
[1.        , 2.54545455],
[1.        , 2.58585859],
[1.        , 2.62626263],
[1.        , 2.66666667],
[1.        , 2.70707071],
[1.        , 2.74747475],
[1.        , 2.78787879],
[1.        , 2.82828283],
[1.        , 2.86868687],
[1.        , 2.90909091],
[1.        , 2.94949495],
[1.        , 2.98989899],
[1.        , 3.03030303],
[1.        , 3.07070707],
[1.        , 3.11111111],
```

```
       [1.        , 3.15151515],
       [1.        , 3.19191919],
       [1.        , 3.23232323],
       [1.        , 3.27272727],
       [1.        , 3.31313131],
       [1.        , 3.35353535],
       [1.        , 3.39393939],
       [1.        , 3.43434343],
       [1.        , 3.47474747],
       [1.        , 3.51515152],
       [1.        , 3.55555556],
       [1.        , 3.5959596 ],
       [1.        , 3.63636364],
       [1.        , 3.67676768],
       [1.        , 3.71717172],
       [1.        , 3.75757576],
       [1.        , 3.7979798 ],
       [1.        , 3.83838384],
       [1.        , 3.87878788],
       [1.        , 3.91919192],
       [1.        , 3.95959596],
       [1.        , 4.        ]])
```

[67]:
```
theta = np.zeros(2)
theta
```

[67]:
```
array([0., 0.])
```

[68]:
```
def compute_cost(X, y, theta):
  """
  Compute cost for linear regression.

  Input Parameters
  ----------------
  X : 2D array where each row represent the training example and each column␣
↪represent the feature ndarray. Dimension(m x n)
      m= number of training examples
      n= number of features (including X_0 column of ones)
  y : 1D array of labels/target value for each traing example. dimension(1 x m)

  theta : 1D array of fitting parameters or weights. Dimension (1 x n)

  Output Parameters
  -----------------
  J : Scalar value.
  """
  theta = np.transpose([data[]])
```

```
  predictions = X.dot(theta)
  #print('predictions= ', predictions[:5])
  errors = np.subtract(predictions, y)
  #print('errors= ', errors[:5])
  sqrErrors = np.square(errors)
  #print('sqrErrors= ', sqrErrors[:5])
  J = 1 / (2 * m) * np.sum(sqrErrors)

  return J
```

```
  File "/tmp/ipykernel_6873/3471724458.py", line 18
    theta = np.transpose([data[]])
                                  ^
SyntaxError: invalid syntax
```

[ ]:
```
# Lets compute the cost for theta values
cost = compute_cost(x1, y, theta)
print('The cost for given values of theta_0 and theta_1 =', cost)
```

[ ]:
```

```