Predict Profit for food truck

Problem statement :

In this project, we are implementing linear regression with one variable to predict profits for a food truck.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. Dataset is like below :

Population (10,000s)	Profit (10,000s $)
5.5277	13.662
8.5186	9.1302
6.1101	6.8233

Now we have to predict profit for given population city(including which is not traverse by our food truck).

Note:This problem statement and dataset is from coursera Andrew ng machine learning Coursework (https://www.coursera.org/learn/machine-learning)

In [2]:
```
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
import pandas as pd  #to work on the datasets pip install pandas

```

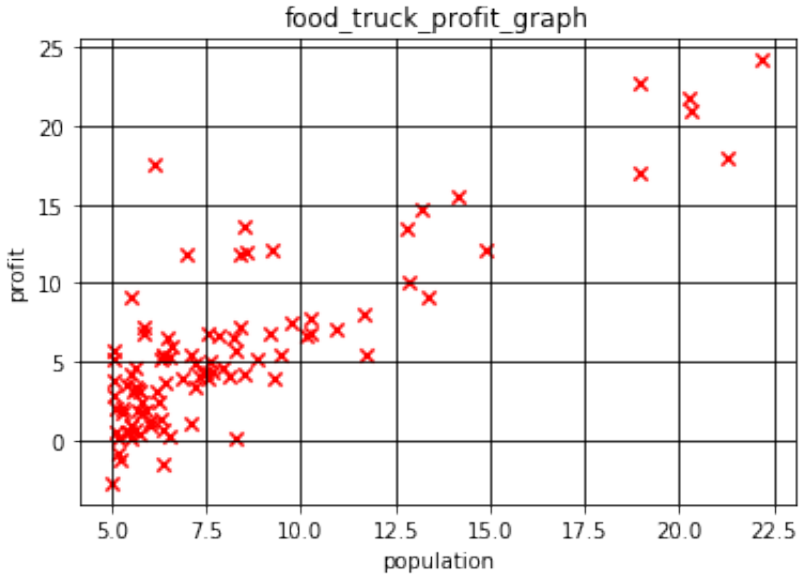
In [3]:
```
#reading data
data=pd.read_csv("profitdataset.txt")
print(data.shape)

```

```
(97, 2)
```

In [4]:
```
# get x and y data values
x=data['population'].values
y=data['profit'].values
```

In [5]:
```
#plot the data values
plt.scatter(x,y,40,c='r',label='scatter_data',marker='x')
plt.xlabel("population")
plt.ylabel('profit')
plt.title('food_truck_profit_graph')
plt.grid(True,color='k')
plt.show()
```



In [6]:
```
#  mean values
mean_x=np.mean(x)
mean_y=np.mean(y)
n=len(x)
mean_x
```

Out[6]: 8.159799999999999

In [7]:
```
#y=mx+c. to get the values m and c
num=0
den=0
for i in range(n):
    num=num+((x[i]-mean_x)*(y[i]-mean_y))
    den=den+((x[i]-mean_x)**2)
m=num/den
#c=y-m*x
c=mean_y-(m*mean_x)
print(m,c)

```

```
1.193033644189594 -3.8957808783118537
```

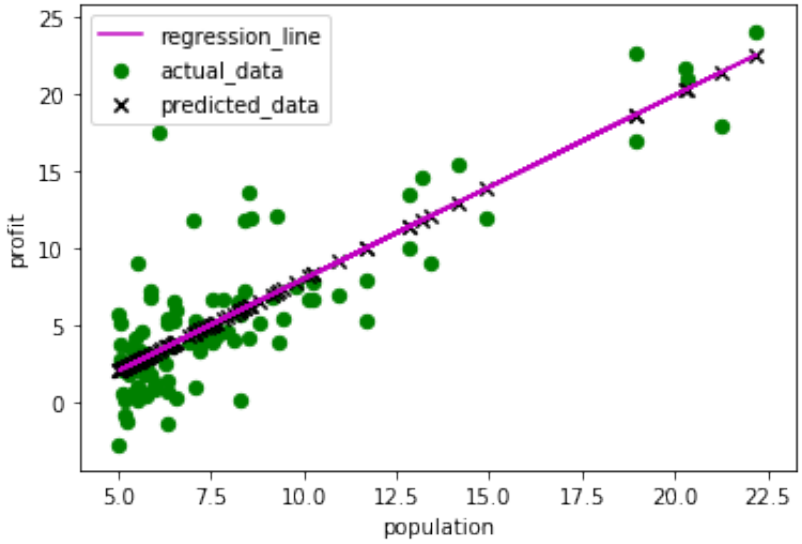
In [8]:
```
#calculate y_predicate
y_pred=np.array([])    #to create empty array

for i in range(n):
    p=(m*x[i])+c    #y=mx+C
    y_pred=np.append(y_pred,np.array([p]),axis=0)

print(y_pred)
```

```
[ 3.39377399  2.6989512   6.26719552  4.45927234  3.09515767  6.1053
0086
  5.02381586  6.33818102  3.84247394  2.13452698  2.91727635 13.0023
4766
  2.94507404  6.13572322  2.833764    2.52202431  3.69835548  2.2246
0102
  3.77494824  4.53992141  3.48802365 20.28701109  2.65409313  3.6514
6926
  2.74333205 18.70624151 11.40845471  9.17628876 11.82363042 22.5931
4512
  2.37050903  3.96559502  7.13763287  3.13333475  5.90033768  5.5690
3223
  5.7629002   2.79272364 11.41799898  3.68403908  2.55483273  4.3152
7318
 10.07225703  2.99243747  5.43934948  4.56652606  2.1531383   3.0254
8451
 10.06271276  2.71553436  5.09993141  2.43648379  4.96118159  5.1749
7322
  3.65946258  3.69060076  3.58955081  2.83257096  7.21160096  7.3826
8198
  6.63321825  2.28329828 21.49078204 13.88996469 18.72294398  4.7157
7457
  6.0005525   8.3161115   2.66518834 20.37171648  8.19680814  4.8545
2438
  3.2698178   4.72496093  2.10147995  3.91608412  5.09802255  2.1129
3307
  8.36144678  2.19787707  2.93934748  2.29415488  3.68678305  7.7586
0688
  3.87790704  6.26552528  7.05650658  3.26480705  2.69024205  2.1402
5354
  2.91369725  5.21493985  3.10816174  2.43373982  5.99852435 12.0837
1175
  2.59062374]
```

In [9]:
```
#plot the regression line
plt.plot(x,y_pred,color='m',label="regression_line")
#plot the actual data
plt.scatter(x,y,40,c='g',label="actual_data",marker='o')
plt.scatter(x,y_pred,40,c='k',label="predicted_data",marker='x')
plt.xlabel("population")
plt.ylabel('profit')
plt.legend()
plt.show()
```



In [10]:
```
# how good our fit line or regression line is by using r square me

nu=0
de=0
for i in range(n):
    nu=nu+((y_pred[i]-mean_y)**2) #predicted values of y
    de=de+((y[i]-mean_y)**2)  #actual values of y
r2=nu/de
r2



#r square ranges from 0 to 1 if >0.7. then its good fit line
```

Out[10]: 0.70203155378414

In [11]:
```
1 x
```

Out[11]:
```
array([ 6.1101,  5.5277,  8.5186,  7.0032,  5.8598,  8.3829,  7.4764
,
        8.5781,  6.4862,  5.0546,  5.7107, 14.164 ,  5.734 ,  8.4084
,
        5.6407,  5.3794,  6.3654,  5.1301,  6.4296,  7.0708,  6.1891
,
       20.27  ,  5.4901,  6.3261,  5.5649, 18.945 , 12.828 , 10.957
,
       13.176 , 22.203 ,  5.2524,  6.5894,  9.2482,  5.8918,  8.2111
,
        7.9334,  8.0959,  5.6063, 12.836 ,  6.3534,  5.4069,  6.8825
,
       11.708 ,  5.7737,  7.8247,  7.0931,  5.0702,  5.8014, 11.7
,
        5.5416,  7.5402,  5.3077,  7.4239,  7.6031,  6.3328,  6.3589
,
        6.2742,  5.6397,  9.3102,  9.4536,  8.8254,  5.1793, 21.279
,
       14.908 , 18.959 ,  7.2182,  8.2951, 10.236 ,  5.4994, 20.341
,
       10.136 ,  7.3345,  6.0062,  7.2259,  5.0269,  6.5479,  7.5386
,
        5.0365, 10.274 ,  5.1077,  5.7292,  5.1884,  6.3557,  9.7687
,
        6.5159,  8.5172,  9.1802,  6.002 ,  5.5204,  5.0594,  5.7077
,
        7.6366,  5.8707,  5.3054,  8.2934, 13.394 ,  5.4369])
```

In [21]:
```
1 # predict the profit for city with 45000 and 65000 people
2 X_test = np.array([[4.5],[6.5]])
3 y1=(m*X_test[0])+c
4 y2=(m*X_test[1])+c
5 print('profit from 45000 people city is ',y1*10000,'$')
6 print('profit from 65000 people city is ',y2*10000,'$')
7
8
```

```
profit from 45000 people city is  [14728.70520541] $
profit from 65000 people city is  [38589.37808921] $
```

End of the program 1