```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Step 1 : Fetching the data

# df = pd.read_csv("/content/Profit_Dataset.txt")
# df.head()
df = pd.read_csv("/content/Profit_Dataset.txt",header=None);
df.head(10)
df.describe()

# # Step 2 : preprocessing the data

# x = df.iloc[:,0]
# print(x)
# m = x.shape[0]
# print(m)
# x=x.values
# type(x)
# x= x.reshape(m,1)
# x.shape
# x
# y= df.iloc[:,1]
# y= (y.values).reshape(m,1)
# y.shape
# plt.scatter(x,y,marker='x')
# plt.xlabel('Population in lakhs')
# plt.ylabel('profit in thousand rs')
# plt.title('Food truck profit Estimation')

# # step 3 : construct a model

# col1 = np.ones((m,1))
# col1
# x= np.hstack((col1,x))
# print(x)
# Theta = np.zeros((2,1))
# j=0
# alpha = 0.1
# print(Theta)

# #hypothesis
# h = np.dot(x,Theta)
# print(h)
# j = np.sum(np.square(h-y))/(2*m)
# print(j)

# Theta[0]-(alpha/m)*np.sum(h-y)
# Theta[1]-(alpha/m)*np.sum((h-y)*(x[:,1].reshape(m,1)))



# def computeCost(x,y,Theta):
#   m=y.shape[0]
#   h=np.dot(x,Theta)
#   j=np.sum(np.square(h-y))/(2*m)
#   return [h,j]


# def gradientDescent(x,y,Theta,alpha):
#   m=y.shape[0]
#   h=computeCost(x,y,Theta)[0]
#   j=computeCost(x,y,Theta)[1]
#   Theta[0]=Theta[0]-(alpha/m)*np.sum(h-y)
#   Theta[1]=Theta[1]-(alpha/m)*np.sum((h-y)*(x[:,1].reshape(m,1)))
#   return [j,Theta]

# def trainLinearRegression(x,y,alpha,noIter,printIter):
#   Theta = np.zeros((2,1))
#   jHistory =[]
#   for i in range(noIter):
#     j=gradientDescent(x,y,Theta,alpha)[0]
#     jHistory.append(j)
#     if(i % printIter==0):
#       print("iteration =",i)
#       print("cost =",j)

#   plot1 = plt.figure(1)
#   plt.scatter(x[:,1],y,marker='x')
#   plt.plot(x,np.dot(x,Theta))
#   plt.xlabel('# iteration')
#   plt.ylabel("profit in lakh rs")
#   plt.title('profit made by a foodtruck')


#   plot2 = plt.figure(2)
#   plt.plot(list(range(noIter)),jHistory)
#   plt.xlabel("# Iteration")
#   plt.ylabel("J")
#   plt.title("convergence of cost function")

#   plt.show()
#   return Theta

# Theta = trainLinearRegression(x,y,0.001,20000,1000)
```

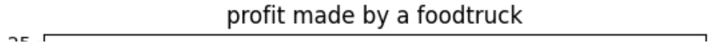
```
0      6.1101
1      5.5277
2      8.5186
3      7.0032
4      5.8598
        ...
92     5.8707
93     5.3054
94     8.2934
95    13.3940
96     5.4369
Name: 0, Length: 97, dtype: float64
97
[[ 1.      6.1101]
 [ 1.      5.5277]
 [ 1.      8.5186]
 [ 1.      7.0032]
 [ 1.      5.8598]
 [ 1.      8.3829]
 [ 1.      7.4764]
 [ 1.      8.5781]
 [ 1.      6.4862]
 [ 1.      5.0546]
 [ 1.      5.7107]
 [ 1.     14.164 ]
 [ 1.      5.734 ]
 [ 1.      8.4084]
 [ 1.      5.6407]
 [ 1.      5.3794]
 [ 1.      6.3654]
 [ 1.      5.1301]
 [ 1.      6.4296]
 [ 1.      7.0708]
 [ 1.      6.1891]
 [ 1.     20.27  ]
 [ 1.      5.4901]
 [ 1.      6.3261]
 [ 1.      5.5649]
 [ 1.     18.945 ]
 [ 1.     12.828 ]
 [ 1.     10.957 ]
 [ 1.     13.176 ]
 [ 1.     22.203 ]
 [ 1.      5.2524]
 [ 1.      6.5894]
 [ 1.      9.2482]
 [ 1.      5.8918]
 [ 1.      8.2111]
 [ 1.      7.9334]
 [ 1.      8.0959]
 [ 1.      5.6063]
 [ 1.     12.836 ]
 [ 1.      6.3534]
 [ 1.      5.4069]
 [ 1.      6.8825]
 [ 1.     11.708 ]
 [ 1.      5.7737]
 [ 1.      7.8247]
```

Double-click (or enter) to edit

```
 [ 1.      5.8014]
```

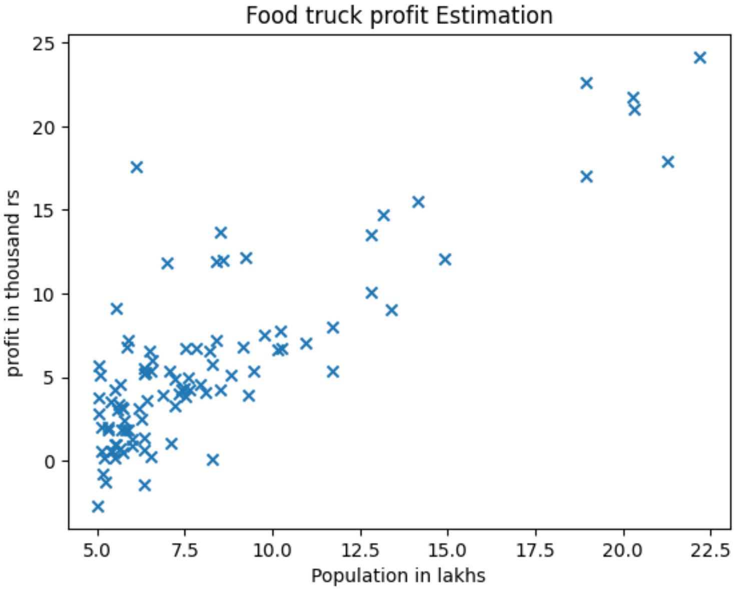
```
# Step 2 : preprocessing the data

x = df.iloc[:,0]
print(x)
m = x.shape[0]
print(m)
x=x.values
type(x)
x= x.reshape(m,1)
x.shape
x
y= df.iloc[:,1]
y= (y.values).reshape(m,1)
y.shape
plt.scatter(x,y,marker='x')
plt.xlabel('Population in lakhs')
plt.ylabel('profit in thousand rs')
plt.title('Food truck profit Estimation')

# step 3 : construct a model

col1 = np.ones((m,1))
col1
x= np.hstack((col1,x))
print(x)
Theta = np.zeros((2,1))
j=0
alpha = 0.1
print(Theta)

#hypothesis
h = np.dot(x,Theta)
print(h)
j = np.sum(np.square(h-y))/(2*m)
print(j)

Theta[0]-(alpha/m)*np.sum(h-y)
Theta[1]-(alpha/m)*np.sum((h-y)*(x[:,1].reshape(m,1)))



def computeCost(x,y,Theta):
  m=y.shape[0]
  h=np.dot(x,Theta)
  j=np.sum(np.square(h-y))/(2*m)
  return [h,j]


def gradientDescent(x,y,Theta,alpha):
  m=y.shape[0]
  h=computeCost(x,y,Theta)[0]
  j=computeCost(x,y,Theta)[1]
  Theta[0]=Theta[0]-(alpha/m)*np.sum(h-y)
  Theta[1]=Theta[1]-(alpha/m)*np.sum((h-y)*(x[:,1].reshape(m,1)))
  return [j,Theta]

def trainLinearRegression(x,y,alpha,noIter,printIter):
  Theta = np.zeros((2,1))
  jHistory =[]
  for i in range(noIter+1):
    j=gradientDescent(x,y,Theta,alpha)[0]
    jHistory.append(j)
    if(i % printIter==0):
      print("iteration =",i)
      print("cost =",j)

  plot1 = plt.figure(1)
  plt.scatter(x[:,1],y,marker='x')
  plt.plot(x,np.dot(x,Theta))
  plt.xlabel('# iteration')
  plt.ylabel("profit in lakh rs")
  plt.title('profit made by a foodtruck')


  plot2 = plt.figure(2)
  plt.plot(list(range(noIter)),jHistory)
  plt.xlabel("# Iteration")
  plt.ylabel("J")
  plt.title("convergence of cost function")

  plt.show()
  return Theta
```

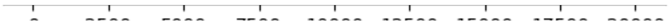
```
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]]
32.072733877455676
iteration = 0
cost = 32.072733877455676
iteration = 1000
cost = 5.480269332020323
iteration = 2000
cost = 5.176562563777922
iteration = 3000
cost = 4.964790400326137
iteration = 4000
cost = 4.817123460031757
iteration = 5000
cost = 4.714156550214695
iteration = 6000
cost = 4.64235859397521
iteration = 7000
cost = 4.592294485716297
iteration = 8000
cost = 4.557385206023154
iteration = 9000
cost = 4.533043260244523
iteration = 10000
cost = 4.516069827120197
iteration = 11000
cost = 4.5042343956125475
iteration = 12000
cost = 4.495981649220805
iteration = 13000
cost = 4.490227078889981
iteration = 14000
cost = 4.486214465619138
iteration = 15000
cost = 4.483416504288282
iteration = 16000
cost = 4.481465509492482
iteration = 17000
cost = 4.48010509730549
iteration = 18000
cost = 4.479156493381733
iteration = 19000
cost = 4.4784950398805305
```



```
Theta = trainLinearRegression(x,y,0.001,20000,1000)
```

```
0      6.1101
1      5.5277
2      8.5186
3      7.0032
4      5.8598
        ...
92     5.8707
93     5.3054
94     8.2934
95    13.3940
96     5.4369
Name: 0, Length: 97, dtype: float64
97
Text(0.5, 1.0, 'Food truck profit Estimation')
```





```
# step 3 : construct a model

col1 = np.ones((m,1))
col1
x= np.hstack((col1,x))
print(x)
Theta = np.zeros((2,1))
j=0
alpha = 0.1
print(Theta)

#hypothesis
h = np.dot(x,Theta)
print(h)
j = np.sum(np.square(h-y))/(2*m)
print(j)

Theta[0]-(alpha/m)*np.sum(h-y)
Theta[1]-(alpha/m)*np.sum((h-y)*(x[:,1].reshape(m,1)))



def computeCost(x,y,Theta):
  m=y.shape[0]
  h=np.dot(x,Theta)
  j=np.sum(np.square(h-y))/(2*m)
  return [h,j]


def gradientDescent(x,y,Theta,alpha):
  m=y.shape[0]
  h=computeCost(x,y,Theta)[0]
  j=computeCost(x,y,Theta)[1]
  Theta[0]=Theta[0]-(alpha/m)*np.sum(h-y)
  Theta[1]=Theta[1]-(alpha/m)*np.sum((h-y)*(x[:,1].reshape(m,1)))
  return [j,Theta]

def trainLinearRegression(x,y,alpha,noIter,printIter):
  Theta = np.zeros((2,1))
  jHistory =[]
  for i in range(noIter+1):
    j=gradientDescent(x,y,Theta,alpha)[0]
    jHistory.append(j)
    if(i % printIter==0):
      print("iteration =",i)
      print("cost =",j)

  plot1 = plt.figure(1)
  plt.scatter(x[:,1],y,marker='x')
  plt.plot(x,np.dot(x,Theta))
  plt.xlabel('# iteration')
  plt.ylabel("profit in lakh rs")
  plt.title('profit made by a foodtruck')


  plot2 = plt.figure(2)
  plt.plot(list(range(noIter)),jHistory)
  plt.xlabel("# Iteration")
  plt.ylabel("J")
  plt.title("convergence of cost function")

  plt.show()
  return Theta

Theta = trainLinearRegression(x,y,0.001,20000,1000)
```

```
def computeCost(x,y,Theta):
  m=y.shape[0]
  h=np.dot(x,Theta)
  j=np.sum(np.square(h-y))/(2*m)
  return [h,j]


def gradientDescent(x,y,Theta,alpha):
  m=y.shape[0]
  h=computeCost(x,y,Theta)[0]
  j=computeCost(x,y,Theta)[1]
  Theta[0]=Theta[0]-(alpha/m)*np.sum(h-y)
  Theta[1]=Theta[1]-(alpha/m)*np.sum((h-y)*(x[:,1].reshape(m,1)))
  return [j,Theta]

def trainLinearRegression(x,y,alpha,noIter,printIter):
  Theta = np.zeros((2,1))
  jHistory =[]
  for i in range(noIter+1):
    j=gradientDescent(x,y,Theta,alpha)[0]
    jHistory.append(j)
    if(i % printIter==0):
      print("iteration =",i)
      print("cost =",j)

  plot1 = plt.figure(1)
```