Linear Regression

Implementing Linear Regression for predicting profit of a food truck given the population and profits of different cities

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

Reading the dataset

In [2]:
```
df = pd.read_csv('data\ex1data1.txt', header = None, names = ['Population', 'Profit'])
df.head()
```

Out[2]:

	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 [3]:
```
df.describe()
```

Out[3]:

	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

In [4]:
```
df.T
```

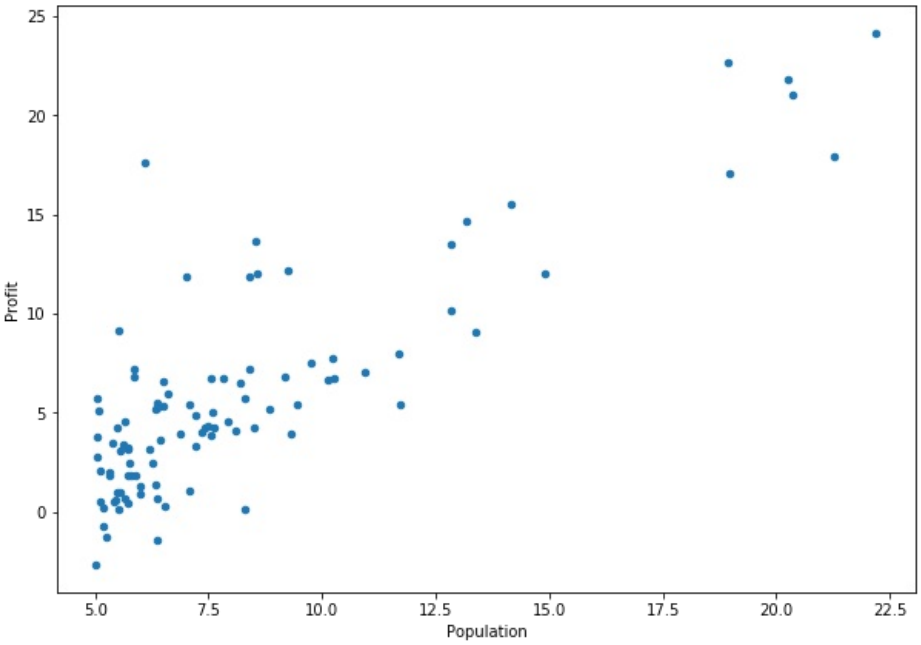
Out[4]:

	0	1	2	3	4	5	6	7	8	9	...	87	88	89	90
Population	6.1101	5.5277	8.5186	7.0032	5.8598	8.3829	7.4764	8.5781	6.4862	5.0546	...	6.00200	5.5204	5.0594	5.7077
Profit	17.5920	9.1302	13.6620	11.8540	6.8233	11.8860	4.3483	12.0000	6.5987	3.8166	...	0.92695	0.1520	2.8214	1.8451

2 rows × 97 columns

In [5]:
```
df.plot(kind = 'scatter', x = 'Population', y = 'Profit', figsize = (10,7))
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b24645e278>
```



Function to compute the cost function

In [6]:
```
def computeCost(X, y, theta):
    inner_product = np.power(((X * theta.T) - y) , 2)
    return np.sum(inner_product) / (2 * len(X))
```

Appending 1s to the front of the dataset

In [7]:
```
df.insert(0, 'Ones', 1)
```

In [8]:
```
df.head()
```

Out[8]:

	Ones	Population	Profit
0	1	6.1101	17.5920
1	1	5.5277	9.1302
2	1	8.5186	13.6620
3	1	7.0032	11.8540
4	1	5.8598	6.8233

Set training data (X) and target variable (y)

In [9]:
```
cols = df.shape[1]
cols
```

Out[9]: 3

In [10]:
```
X = df.iloc[:, 0:cols-1]
y = df.iloc[:, cols-1:cols]
```

Converting dataframe to numpy matrices and creating a parameter (theta) matrix

In [11]:
```
X = np.matrix(X.values)
y = np.matrix(y.values)
```

In [12]:
```
theta = np.matrix(np.array([0,0]))
```

In [13]:
```
theta
```

Out[13]:
```
matrix([[0, 0]])
```

In [14]:
```
X.shape, y.shape, theta.shape
```

Out[14]:
```
((97, 2), (97, 1), (1, 2))
```

In [15]:
```
computeCost(X, y, theta)
```

Out[15]:
```
32.072733877455676
```

Function to perform gradient descent on parameters theta by repeatedly updating the parameters

In [16]:
```
def gradientDescent(X, y, theta, alpha, iters):
    temp = np.matrix(np.zeros(theta.shape))
    parameters = int(theta.ravel().shape[1])
    cost = np.zeros(iters)

    # With each iteration, we are calculating the changes to be made to the
    # parameter (theta) value to reduce the error (cost)
    for i in range(iters):
        error = (X * theta.T) - y

        # number of parameters to be calculated,here we have 2 parameters theta [0,0] and theta [0,1]
        for j in range(parameters):
            term  = np.multiply(error, X[:, j])
            temp[0,j] = theta[0,j] -((alpha / len(X)) * np.sum(term))

        theta = temp
        cost[i] = computeCost(X, y, theta)

    return theta, cost
```

Set values for learning rate and number of iterations

- alpha here denotes the learning rate that helps to determine how quickly the algorithm will converge to the optimal solution
- iters denotes the number of iterations

In [17]:
```
alpha = 0.01
iters = 1000
# perform gradient descent to fit the model parameters and compute the cost
g, cost = gradientDescent(X, y, theta, alpha, iters)
print(g)
print(cost)
```

```
[[-3.24140214  1.1272942 ]]
[ 6.73719046  5.93159357  5.90115471  5.89522859  5.89009494  5.88500416
  5.87993248  5.87487909  5.86984391  5.86482687  5.85982789  5.85484692
  5.84988389  5.84493874  5.8400114   5.83510181  5.8302099   5.82533562
  5.82047889  5.81563965  5.81081784  5.8060134   5.80122627  5.79645638
  5.79170367  5.78696808  5.78224955  5.77754801  5.77286341  5.76819568
  5.76354477  5.75891061  5.75429313  5.7496923   5.74510803  5.74054027
  5.73598897  5.73145406  5.72693549  5.72243319  5.71794711  5.71347718
  5.70902336  5.70458558  5.70016379  5.69575792  5.69136792  5.68699373
  5.6826353   5.67829257  5.67396548  5.66965398  5.665358    5.6610775
  5.65681242  5.6525627   5.64832829  5.64410913  5.63990517  5.63571635
  5.63154261  5.62738391  5.6232402   5.6191114   5.61499748  5.61089837
  5.60681403  5.60274441  5.59868944  5.59464907  5.59062326  5.58661195
  5.58261509  5.57863263  5.57466451  5.57071068  5.56677109  5.5628457
  5.55893444  5.55503727  5.55115414  5.547285    5.54342979  5.53958847
  5.53576098  5.53194728  5.52814732  5.52436105  5.52058841  5.51682936
  5.51308385  5.50935183  5.50563326  5.50192808  5.49823624  5.49455771
  5.49089242  5.48724033  5.4836014   5.47997558  5.47636282  5.47276307
  5.46917628  5.46560242  5.46204143  5.45849326  5.45495788  5.45143522
  5.44792526  5.44442794  5.44094322  5.43747105  5.43401138  5.43056418
  5.42712939  5.42370698  5.42029689  5.41689909  5.41351352  5.41014015
  5.40677893  5.40342982  5.40009277  5.39676774  5.39345469  5.39015357
  5.38686434  5.38358696  5.38032138  5.37706756  5.37382547  5.37059505
  5.36737627  5.36416908  5.36097345  5.35778933  5.35461667  5.35145544
  5.3483056   5.34516711  5.34203991  5.33892399  5.33581928  5.33272576
```

5.32964339 5.32657211 5.3235119 5.32046271 5.3174245 5.31439724
5.31138088 5.30837538 5.30538071 5.30239683 5.2994237 5.29646127
5.29350951 5.29056839 5.28763786 5.28471789 5.28180843 5.27890945
5.27602092 5.27314279 5.27027503 5.26741759 5.26457045 5.26173356
5.2589069 5.25609041 5.25328407 5.25048783 5.24770167 5.24492555
5.24215942 5.23940326 5.23665702 5.23392068 5.23119419 5.22847752
5.22577064 5.22307351 5.22038609 5.21770835 5.21504026 5.21238178
5.20973288 5.20709351 5.20446365 5.20184327 5.19923232 5.19663078
5.19403861 5.19145577 5.18888224 5.18631798 5.18376295 5.18121713
5.17868048 5.17615296 5.17363455 5.17112521 5.16862491 5.16613361
5.16365129 5.16117791 5.15871344 5.15625784 5.15381109 5.15137315
5.148944 5.14652359 5.1441119 5.1417089 5.13931455 5.13692883
5.1345517 5.13218314 5.1298231 5.12747157 5.1251285 5.12279388
5.12046766 5.11814983 5.11584034 5.11353917 5.11124629 5.10896167
5.10668527 5.10441708 5.10215706 5.09990518 5.0976614 5.09542571
5.09319808 5.09097846 5.08876685 5.08656319 5.08436748 5.08217967
5.07999975 5.07782768 5.07566343 5.07350697 5.07135828 5.06921734
5.0670841 5.06495855 5.06284065 5.06073039 5.05862772 5.05653263
5.05444508 5.05236505 5.05029252 5.04822745 5.04616982 5.0441196
5.04207676 5.04004129 5.03801314 5.0359923 5.03397874 5.03197243
5.02997335 5.02798147 5.02599676 5.0240192 5.02204877 5.02008543
5.01812917 5.01617995 5.01423775 5.01230255 5.01037431 5.00845303
5.00653866 5.00463119 5.00273059 5.00083684 4.9989499 4.99706977
4.9951964 4.99332979 4.99146989 4.9896167 4.98777018 4.98593031
4.98409707 4.98227043 4.98045038 4.97863687 4.9768299 4.97502944
4.97323547 4.97144795 4.96966687 4.96789221 4.96612395 4.96436205
4.96260649 4.96085726 4.95911433 4.95737768 4.95564729 4.95392312
4.95220517 4.95049341 4.94878781 4.94708835 4.94539502 4.94370778
4.94202663 4.94035152 4.93868246 4.9370194 4.93536233 4.93371123
4.93206608 4.93042686 4.92879354 4.9271661 4.92554453 4.92392879
4.92231888 4.92071476 4.91911642 4.91752384 4.915937 4.91435587
4.91278043 4.91121067 4.90964657 4.9080881 4.90653524 4.90498798
4.90344629 4.90191015 4.90037954 4.89885445 4.89733485 4.89582073
4.89431206 4.89280882 4.891311 4.88981857 4.88833152 4.88684982
4.88537346 4.88390242 4.88243668 4.88097622 4.87952102 4.87807106
4.87662632 4.87518678 4.87375243 4.87232325 4.87089922 4.86948031
4.86806652 4.86665782 4.86525419 4.86385562 4.86246208 4.86107357
4.85969005 4.85831152 4.85693796 4.85556934 4.85420565 4.85284687
4.85149299 4.85014399 4.84879984 4.84746054 4.84612606 4.84479638
4.8434715 4.84215138 4.84083603 4.83952541 4.83821951 4.83691831
4.83562181 4.83432997 4.83304278 4.83176023 4.83048231 4.82920898
4.82794024 4.82667607 4.82541645 4.82416138 4.82291082 4.82166476
4.8204232 4.8191861 4.81795347 4.81672527 4.81550149 4.81428213
4.81306715 4.81185656 4.81065032 4.80944843 4.80825086 4.80705761
4.80586866 4.80468399 4.80350359 4.80232744 4.80115553 4.79998784
4.79882435 4.79766505 4.79650993 4.79535897 4.79421216 4.79306948
4.79193091 4.79079644 4.78966606 4.78853976 4.7874175 4.7862993
4.78518511 4.78407495 4.78296878 4.78186659 4.78076838 4.77967412
4.7785838 4.77749741 4.77641493 4.77533635 4.77426166 4.77319084
4.77212387 4.77106075 4.77000146 4.76894598 4.7678943 4.76684642
4.7658023 4.76476195 4.76372535 4.76269247 4.76166332 4.76063788
4.75961613 4.75859806 4.75758366 4.75657291 4.7555658 4.75456232
4.75356245 4.75256619 4.75157351 4.75058441 4.74959887 4.74861688
4.74763843 4.7466635 4.74569209 4.74472417 4.74375974 4.74279878
4.74184129 4.74088724 4.73993663 4.73898945 4.73804567 4.7371053
4.73616831 4.7352347 4.73430445 4.73337755 4.73245399 4.73153376
4.73061684 4.72970322 4.7287929 4.72788585 4.72698207 4.72608155
4.72518427 4.72429022 4.72339939 4.72251177 4.72162735 4.72074611
4.71986805 4.71899315 4.7181214 4.71725279 4.71638731 4.71552495
4.71466569 4.71380953 4.71295645 4.71210645 4.71125951 4.71041561
4.70957476 4.70873694 4.70790213 4.70707033 4.70624153 4.70541571
4.70459287 4.70377299 4.70295606 4.70214208 4.70133103 4.7005229
4.69971768 4.69891536 4.69811593 4.69731938 4.6965257 4.69573487
4.6949469 4.69416177 4.69337946 4.69259997 4.69182329 4.6910494
4.69027831 4.68950999 4.68874444 4.68798164 4.68722159 4.68646428
4.6857097 4.68495784 4.68420868 4.68346223 4.68271846 4.68197737
4.68123895 4.68050319 4.67977008 4.67903961 4.67831177 4.67758656
4.67686395 4.67614395 4.67542654 4.67471172 4.67399947 4.67328979
4.67258266 4.67187808 4.67117604 4.67047652 4.66977953 4.66908504
4.66839306 4.66770357 4.66701656 4.66633203 4.66564997 4.66497036
4.6642932 4.66361847 4.66294618 4.66227631 4.66160885 4.6609438
4.66028114 4.65962087 4.65896298 4.65830745 4.65765429 4.65700348
4.65635502 4.65570889 4.65506509 4.6544236 4.65378443 4.65314756
4.65251298 4.65188069 4.65125068 4.65062293 4.64999745 4.64937422
4.64875324 4.64813449 4.64751797 4.64690367 4.64629158 4.6456817
4.64507402 4.64446852 4.64386521 4.64326406 4.64266509 4.64206827
4.6414736 4.64088107 4.64029068 4.63970241 4.63911626 4.63853223
4.63795029 4.63737046 4.63679271 4.63621704 4.63564345 4.63507192
4.63450245 4.63393503 4.63336966 4.63280632 4.63224501 4.63168573
4.63112845 4.63057319 4.63001992 4.62946865 4.62891937 4.62837206
4.62782672 4.62728335 4.62674193 4.62620247 4.62566495 4.62512936
4.62459571 4.62406397 4.62353415 4.62300624 4.62248023 4.62195612
4.62143389 4.62091355 4.62039507 4.61987847 4.61936373 4.61885084

```
4.6183398   4.6178306   4.61732323  4.61681769  4.61631397  4.61581207
4.61531197  4.61481368  4.61431718  4.61382246  4.61332954  4.61283838
4.612349    4.61186137  4.61137551  4.61089139  4.61040902  4.60992838
4.60944948  4.6089723   4.60849684  4.6080231   4.60755106  4.60708071
4.60661207  4.60614511  4.60567983  4.60521623  4.6047543   4.60429404
4.60383543  4.60337847  4.60292316  4.60246949  4.60201745  4.60156704
4.60111826  4.60067109  4.60022553  4.59978157  4.59933922  4.59889846
4.59845928  4.59802169  4.59758567  4.59715123  4.59671835  4.59628702
4.59585726  4.59542904  4.59500236  4.59457722  4.59415361  4.59373152
4.59331096  4.59289191  4.59247437  4.59205834  4.5916438   4.59123076
4.5908192   4.59040913  4.59000053  4.58959341  4.58918775  4.58878355
4.58838081  4.58797952  4.58757968  4.58718127  4.5867843   4.58638876
4.58599465  4.58560195  4.58521067  4.5848208   4.58443233  4.58404526
4.58365959  4.58327531  4.5828924   4.58251088  4.58213074  4.58175196
4.58137454  4.58099849  4.58062379  4.58025044  4.57987844  4.57950777
4.57913844  4.57877044  4.57840377  4.57803841  4.57767437  4.57731165
4.57695023  4.57659011  4.57623129  4.57587376  4.57551752  4.57516256
4.57480888  4.57445648  4.57410534  4.57375547  4.57340686  4.57305951
4.5727134   4.57236854  4.57202493  4.57168255  4.57134141  4.57100149
4.5706628   4.57032533  4.56998908  4.56965403  4.56932019  4.56898756
4.56865612  4.56832588  4.56799682  4.56766896  4.56734227  4.56701676
4.56669242  4.56636925  4.56604724  4.5657264   4.56540671  4.56508817
4.56477078  4.56445453  4.56413942  4.56382544  4.5635126   4.56320088
4.56289029  4.56258082  4.56227246  4.56196521  4.56165906  4.56135402
4.56105008  4.56074723  4.56044548  4.56014481  4.55984522  4.55954672
4.55924929  4.55895293  4.55865763  4.5583634   4.55807023  4.55777812
4.55748706  4.55719705  4.55690808  4.55662015  4.55633326  4.5560474
4.55576258  4.55547878  4.555196    4.55491424  4.55463349  4.55435376
4.55407503  4.55379731  4.55352058  4.55324486  4.55297012  4.55269638
4.55242362  4.55215185  4.55188105  4.55161123  4.55134239  4.55107451
4.55080759  4.55054164  4.55027664  4.5500126   4.54974951  4.54948737
4.54922617  4.54896591  4.54870659  4.5484482   4.54819075  4.54793422
4.54767862  4.54742393  4.54717017  4.54691731  4.54666537  4.54641434
4.54616421  4.54591498  4.54566665  4.54541921  4.54517267  4.54492701
4.54468224  4.54443835  4.54419534  4.5439532   4.54371194  4.54347154
4.54323201  4.54299334  4.54275554  4.54251859  4.54228249  4.54204724
4.54181284  4.54157929  4.54134657  4.5411147   4.54088366  4.54065345
4.54042407  4.54019552  4.53996779  4.53974088  4.53951478  4.53928951
4.53906504  4.53884138  4.53861853  4.53839648  4.53817523  4.53795478
4.53773512  4.53751625  4.53729817  4.53708088  4.53686437  4.53664864
4.53643368  4.5362195   4.5360061   4.53579346  4.53558158  4.53537047
4.53516012  4.53495053  4.53474169  4.53453361  4.53432627  4.53411968
4.53391384  4.53370873  4.53350437  4.53330074  4.53309784  4.53289568
4.53269424  4.53249353  4.53229355  4.53209428  4.53189573  4.53169789
4.53150077  4.53130436  4.53110866  4.53091366  4.53071936  4.53052576
4.53033286  4.53014066  4.52994915  4.52975832  4.52956819  4.52937874
4.52918997  4.52900188  4.52881447  4.52862773  4.52844167  4.52825628
4.52807155  4.52788749  4.52770409  4.52752136  4.52733928  4.52715786
4.52697709  4.52679697  4.5266175   4.52643868  4.5262605   4.52608297
4.52590607  4.52572981  4.52555418  4.52537919  4.52520483  4.52503109
4.52485799  4.5246855   4.52451364  4.52434239  4.52417177  4.52400175
4.52383235  4.52366356  4.52349538  4.5233278   4.52316083  4.52299446
4.52282868  4.52266351  4.52249893  4.52233494  4.52217154  4.52200874
4.52184651  4.52168488  4.52152382  4.52136335  4.52120345  4.52104413
4.52088538  4.5207272   4.5205696   4.52041256  4.52025609  4.52010018
4.51994483  4.51979004  4.51963581  4.51948214  4.51932902  4.51917645
4.51902443  4.51887295  4.51872203  4.51857164  4.5184218   4.5182725
4.51812373  4.51797551  4.51782781  4.51768065  4.51753402  4.51738791
4.51724233  4.51709728  4.51695275  4.51680874  4.51666525  4.51652227
4.51637981  4.51623786  4.51609643  4.5159555 ]
```

In [18]:
```
computeCost(X, y, g)
```

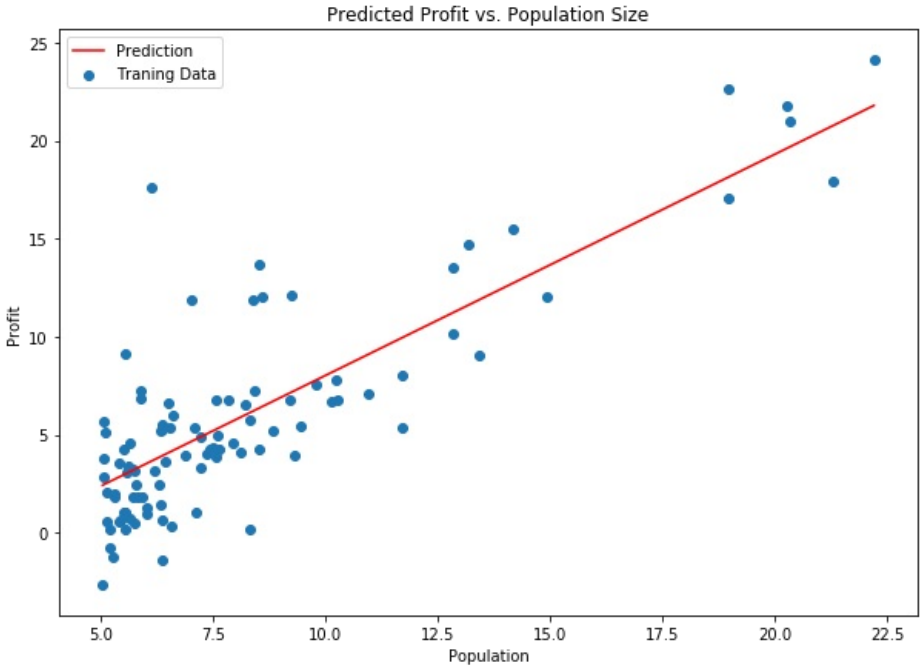
Out[18]:
```
4.5159555030789118
```

Using Matplotlib library to visualize our solution using scatter plot

In [19]:
```
# linspace function will create an evenly-spaced series of points within the range of our data
x = np.linspace(df.Population.min(), df.Population.max(), 100)
f = g[0, 0] + (g[0, 1] * x)

fig, ax = plt.subplots(figsize=(10, 7))
ax.plot(x, f, 'r', label='Prediction')
ax.scatter(df.Population, df.Profit, label='Traning Data')
ax.legend(loc=2)
ax.set_xlabel('Population')
ax.set_ylabel('Profit')
ax.set_title('Predicted Profit vs. Population Size')
```

Out[19]:
```
Text(0.5,1,'Predicted Profit vs. Population Size')
```



Plotting the graph of the gradient descent function which outputs a vector with the cost at each training iteration.

In [20]:
```
fig, ax = plt.subplots(figsize = (12, 8))
ax.plot(np.arange(iters), cost, 'r')
ax.set_xlabel('Iterations')
ax.set_ylabel('Cost')
ax.set_title('Error vs Training epoch')
```

Out[20]:
```
Text(0.5,1,'Error vs Training epoch')
```