LINEAR-REGRESSION

import random

import matplotlib.pyplot as plt

from utils import draw\_line, draw\_plot

def generate\_points(n):

interval = 2

points = ([], [])

max = interval

for x in range(n):

x = random.randint(max - interval, max)

y = random.randint(max - interval, max)

points[0].append(x)

points[1].append(y)

max += interval

return points

L\_RATE = 0.0000001

theta0 = 10

theta1 = 10

def h(x):

return theta0 + theta1 \* x

def cost\_func(x, y):

size = len(x)

# cost is sum((h(Xi) - Yi))^2 / 2m

cost = 0

for i in range(size):

cost += (h(x[i]) - y[i]) \*\* 2

cost /= (2 \* size)

return cost

def update\_parameters(x, y):

global theta0, theta1

deviation0 = 0

deviation1 = 0

size = len(x)

for i in range(size):

deviation0 += (h(x[i]) - y[i])

deviation1 += ((h(x[i]) - y[i]) \* x[i])

deviation0 = (deviation0 \* L\_RATE) / size

deviation1 = (deviation1 \* L\_RATE) / size

theta0 -= deviation0

theta1 -= deviation1

iterations = 10000

cost\_list = [[], []]

x, y = generate\_points(50)

for i in range(iterations):

update\_parameters(x, y)

if(i % 100 == 0):

cost\_list[0].append(i)

cost\_list[1].append(cost\_func(x, y))

print('cost:',cost\_func(x,y))

# Draw plot of cost vs number of iterations

# iterations, cost = cost\_list

# draw\_plot(plt, iterations, 'Iterations', cost, 'Cost', 'line')

# plt.show()

# To show the fitted line and data points

draw\_plot(plt, x, 'Size of house', y, 'Cost of house', 'dot')

xs = [0, max(x)]

ys = [h(xs[0]), h(xs[1])]

draw\_line(plt, xs, ys)

plt.show()

output:

RESTART: C:\Users\dell store\Desktop\ml-workshop-master\linear-regression.py

cost: 139080.22400295208

cost: 130130.65683431881

cost: 121757.03293489793

cost: 113922.28786956453

cost: 106591.74245956293

cost: 99732.94928096328

cost: 93315.5490416091

cost: 87311.13620082328

cost: 81693.1332370683

cost: 76436.67300702556

cost: 71518.48867537486

cost: 66916.81072806739

cost: 62611.270613239954

cost: 58582.810583251885

cost: 54813.59933877567

cost: 51286.95310155485

cost: 47987.26176647014

cost: 44899.919806039725

cost: 42011.261621512065

cost: 39308.50105439539

cost: 36779.67479068168

cost: 34413.58940725243

cost: 32199.771826077347

cost: 30128.422956898747

cost: 28190.37432320965

cost: 26377.047479537112

cost: 24680.416040398602

cost: 23092.970152859096

cost: 21607.6832554321

cost: 20217.980976189174

cost: 18917.712032410276

cost: 17701.121002967095

cost: 16562.822852921217

cost: 15497.779097574023

cost: 14501.275500463216

cost: 13568.901206589378

cost: 12696.529218509344

cost: 11880.298128877417

cost: 11116.595028576285

cost: 10402.039514783537

cost: 9733.468728188127

cost: 9107.923353126396

cost: 8522.634518669978

cost: 7975.011542685093

cost: 7462.630464614585

cost: 6983.223316224837

cost: 6534.668082826581

cost: 6114.979310534331

cost: 5722.299317989281

cost: 5354.889973645752

cost: 5011.125002224775

cost: 4689.482786280474

cost: 4388.539631016899

cost: 4106.963462542835

cost: 3843.5079316711535

cost: 3597.0068971641176

cost: 3366.369264005654

cost: 3150.574153853108

cost: 2948.6663862912474

cost: 2759.752250887099

cost: 2582.9955513312498

cost: 2417.6139041558104

cost: 2262.875275645796

cost: 2118.0947416152117

cost: 1982.6314557055189

cost: 1855.885812787238

cost: 1737.2967949088506

cost: 1626.3394880453425

cost: 1522.5227586547312

cost: 1425.3870797581662

cost: 1334.502496921188

cost: 1249.46672513287

cost: 1169.9033681589844

cost: 1095.460252487488

cost: 1025.8078684918112

cost: 960.6379119120026

cost: 899.6619191978683

cost: 842.6099906736671

cost: 789.2295958726734

cost: 739.2844557536085

cost: 692.553496851265

cost: 648.8298727320736

cost: 607.9200484232073

cost: 569.6429437626263

cost: 533.8291318782581

cost: 500.3200892484892

cost: 468.96749402449944

cost: 439.6325695085971

cost: 412.18546988253706

cost: 386.5047054668939

cost: 362.4766049674543

cost: 339.99481232839275

cost: 318.95981596510376

cost: 299.2785082929347

cost: 280.863773602163

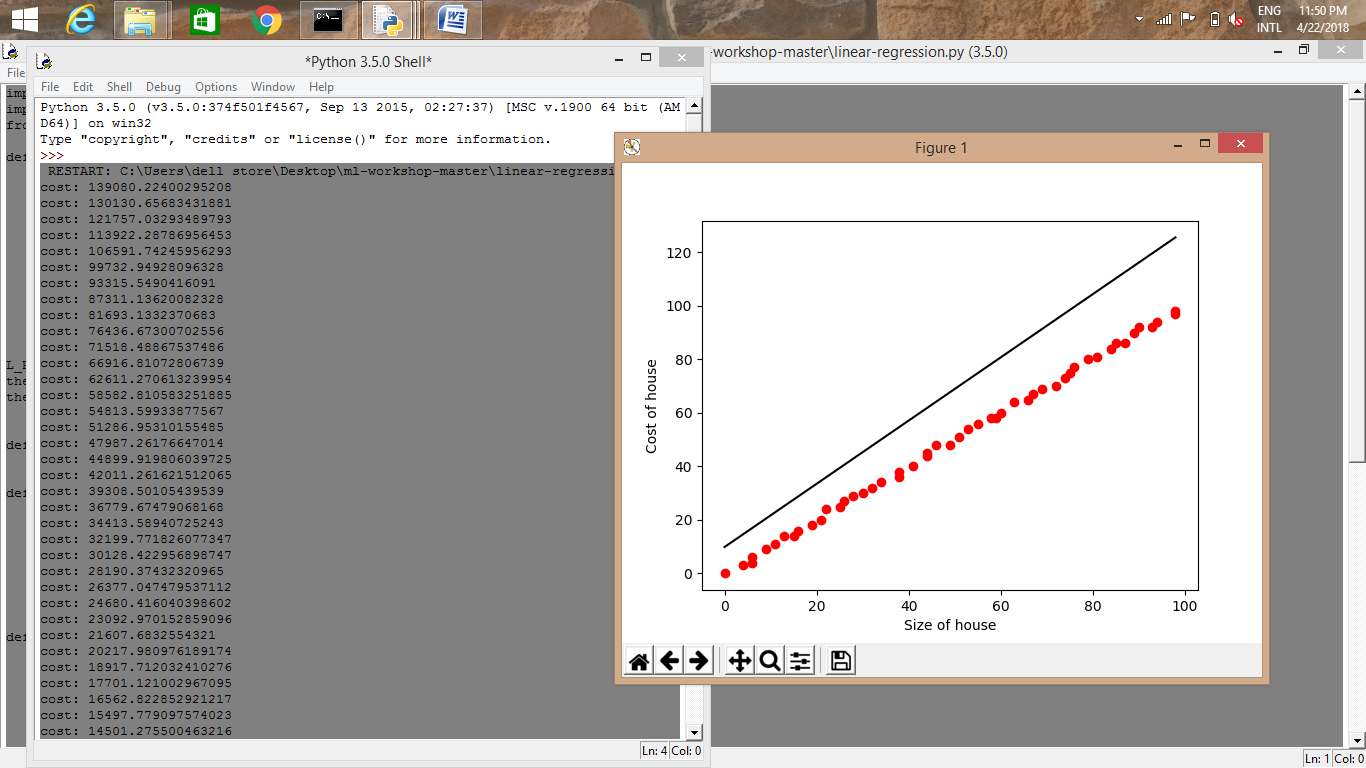
cost: 263.6341024549707

cost: 247.5132308976532

cost: 232.42980289108388

cost: 218.3170544652208

cost: 205.11251819965003



MULTIVARIENT

1.AIR POLLUTION

import numpy as np

import csv

import matplotlib.pyplot as plt

n = 16 # number of input features.

m = 60 # number of training examples.

grad = np.zeros(shape = (n, 1))

theta = np.ones(shape=(n, 1), dtype = float)

hx = np.ones(shape=(m, 1), dtype = float)

file\_handle = open("datasets/air-pollution/data.csv", "r")

reader = csv.reader(file\_handle, delimiter = ',')

learning\_rate = 1e-6

def h(X):

global theta

res = np.matmul(np.transpose(theta), X)

return res

cost\_list = []

itr\_list = []

def gradient\_descent\_algorithm():

global theta, grad

num\_itrs = 100

for itr in range(num\_itrs):

file\_handle.seek(0)

total\_cost = 0.0

idx = 0

for row in reader:

X = [float(x) for x in row[0: -1]]

# list of floats

X = np.asarray(X)

np.reshape(X, [n, 1])

hx[idx][0] = h(X)

y\_correct = float(row[0])

diff = (hx[idx][0] - y\_correct)

total\_cost += (diff \* diff)

idx += 1

for j in range(n):

grad[j][0] = 0.0

i = 0

file\_handle.seek(0)

for row in reader:

y\_correct = float(row[-1])

xij = float(row[j + 1])

diff = hx[i][0] - y\_correct

grad[j][0] += ((learning\_rate \* diff \* xij) / m)

i += 1

theta = theta - grad

total\_cost = total\_cost /(2 \* m)

cost\_list.append(total\_cost)

itr\_list.append(itr + 1)

gradient\_descent\_algorithm()

plt.plot(itr\_list, cost\_list, label = "cost")

plt.xlabel("iterations")

# naming the y axis

plt.ylabel('Cost')

# giving a title to my graph

plt.title('Cost vs iterations')

# show a legend on the plot

plt.legend()

# function to show the plot

plt.show()

ypaxis = []

ycaxis = []

xaxis = []

index = 0

file\_handle.seek(0)

for row in reader:

X = [float(x) for x in row[1:]]

# list of floats

X = np.asarray(X)

np.reshape(X, [n, 1])

pred = h(X)

y\_correct = float(row[0])

index += 1

ypaxis.append(pred)

ycaxis.append(y\_correct)

xaxis.append(index)

plt.plot(xaxis, ycaxis, label = "correct")

plt.plot(xaxis, ypaxis, label = "prediction")

plt.xlabel("examples")

# naming the y axis

plt.ylabel('h\_theta')

plt.title('correct vs predicted')

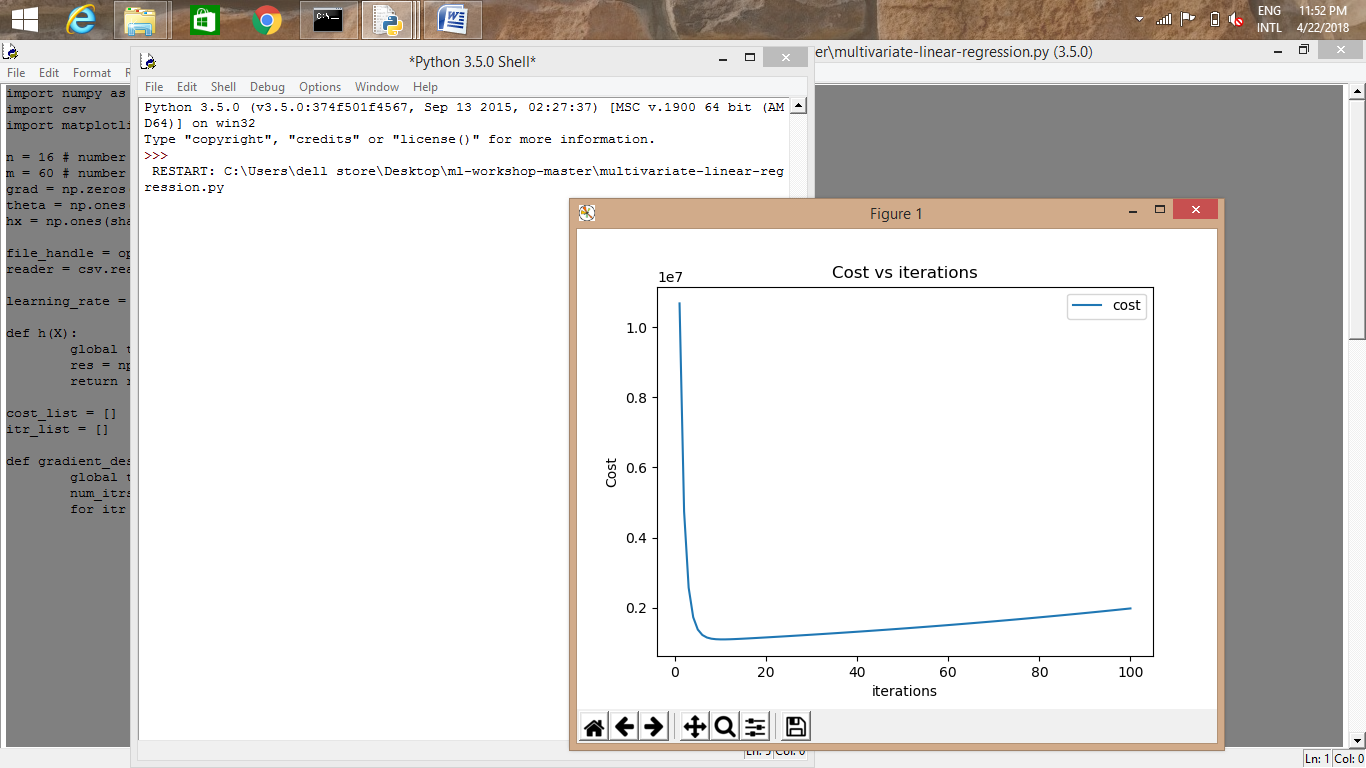
# show a legend on the plot

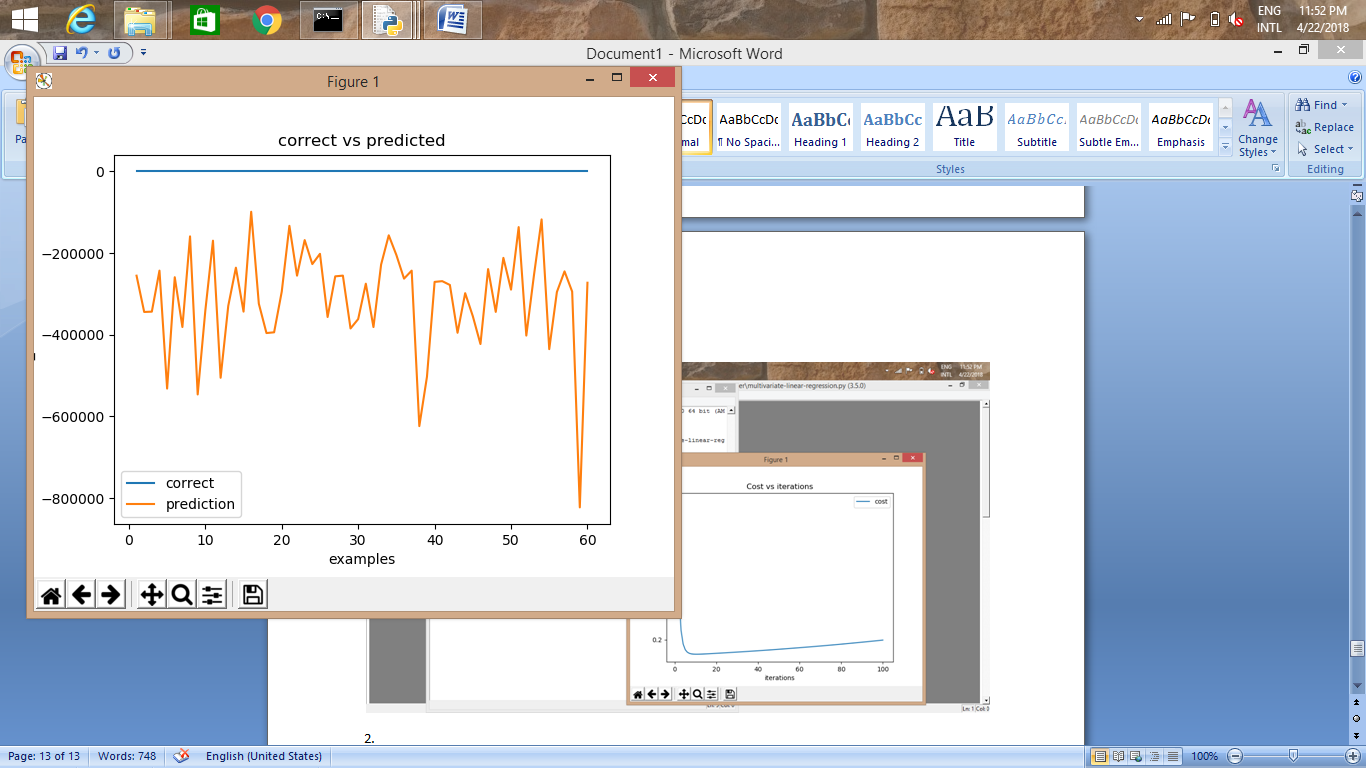
plt.legend()

# function to show the plot

plt.show()

OUTPUT:





2.DEGREE OF ILLNESS

import numpy as np

import csv

import matplotlib.pyplot as plt

n = 3 # number of input features.

m = 53 # number of training examples.

grad = np.zeros(shape = (n, 1))

theta = np.ones(shape=(n, 1), dtype = float)

hx = np.ones(shape=(m, 1), dtype = float)

file\_handle = open("datasets/degree-of-illness/data.csv", "r")

reader = csv.reader(file\_handle, delimiter = ',')

learning\_rate = 1e-6

def h(X):

global theta

res = np.matmul(np.transpose(theta), X)

return res

cost\_list = []

itr\_list = []

def gradient\_descent\_algorithm():

global theta, grad

num\_itrs = 100

for itr in range(num\_itrs):

file\_handle.seek(0)

total\_cost = 0.0

idx = 0

for row in reader:

X = [float(x) for x in row[0: -1]]

# list of floats

X = np.asarray(X)

np.reshape(X, [n, 1])

hx[idx][0] = h(X)

y\_correct = float(row[0])

diff = (hx[idx][0] - y\_correct)

total\_cost += (diff \* diff)

idx += 1

for j in range(n):

grad[j][0] = 0.0

i = 0

file\_handle.seek(0)

for row in reader:

y\_correct = float(row[-1])

xij = float(row[j + 1])

diff = hx[i][0] - y\_correct

grad[j][0] += ((learning\_rate \* diff \* xij) / m)

i += 1

theta = theta - grad

total\_cost = total\_cost /(2 \* m)

cost\_list.append(total\_cost)

itr\_list.append(itr + 1)

gradient\_descent\_algorithm()

plt.plot(itr\_list, cost\_list, label = "cost")

plt.xlabel("iterations")

# naming the y axis

plt.ylabel('Cost')

# giving a title to my graph

plt.title('Cost vs iterations')

# show a legend on the plot

plt.legend()

# function to show the plot

plt.show()

ypaxis = []

ycaxis = []

xaxis = []

index = 0

file\_handle.seek(0)

for row in reader:

X = [float(x) for x in row[1:]]

# list of floats

X = np.asarray(X)

np.reshape(X, [n, 1])

pred = h(X)

y\_correct = float(row[0])

index += 1

ypaxis.append(pred)

ycaxis.append(y\_correct)

xaxis.append(index)

plt.plot(xaxis, ycaxis, label = "correct")

plt.plot(xaxis, ypaxis, label = "prediction")

plt.xlabel("examples")

# naming the y axis

plt.ylabel('h\_theta')

plt.title('correct vs predicted')

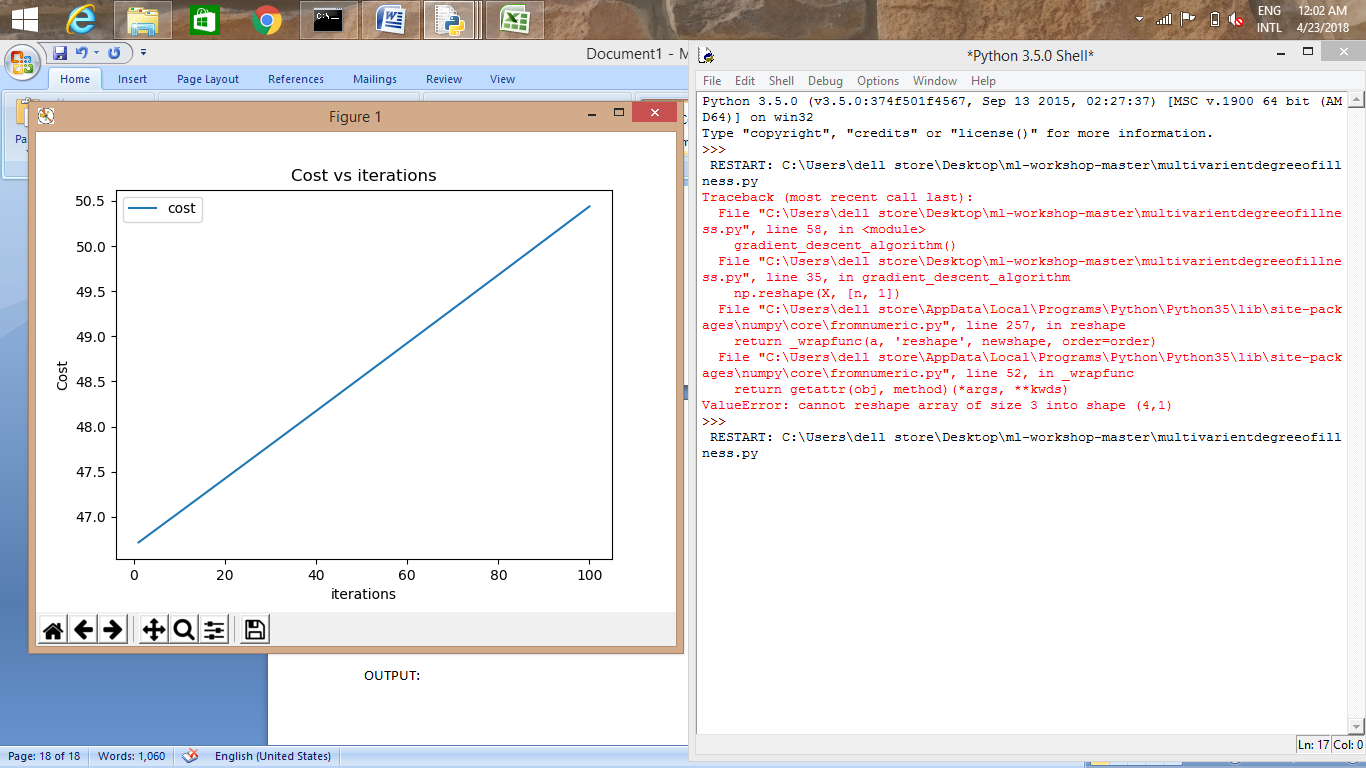
# show a legend on the plot

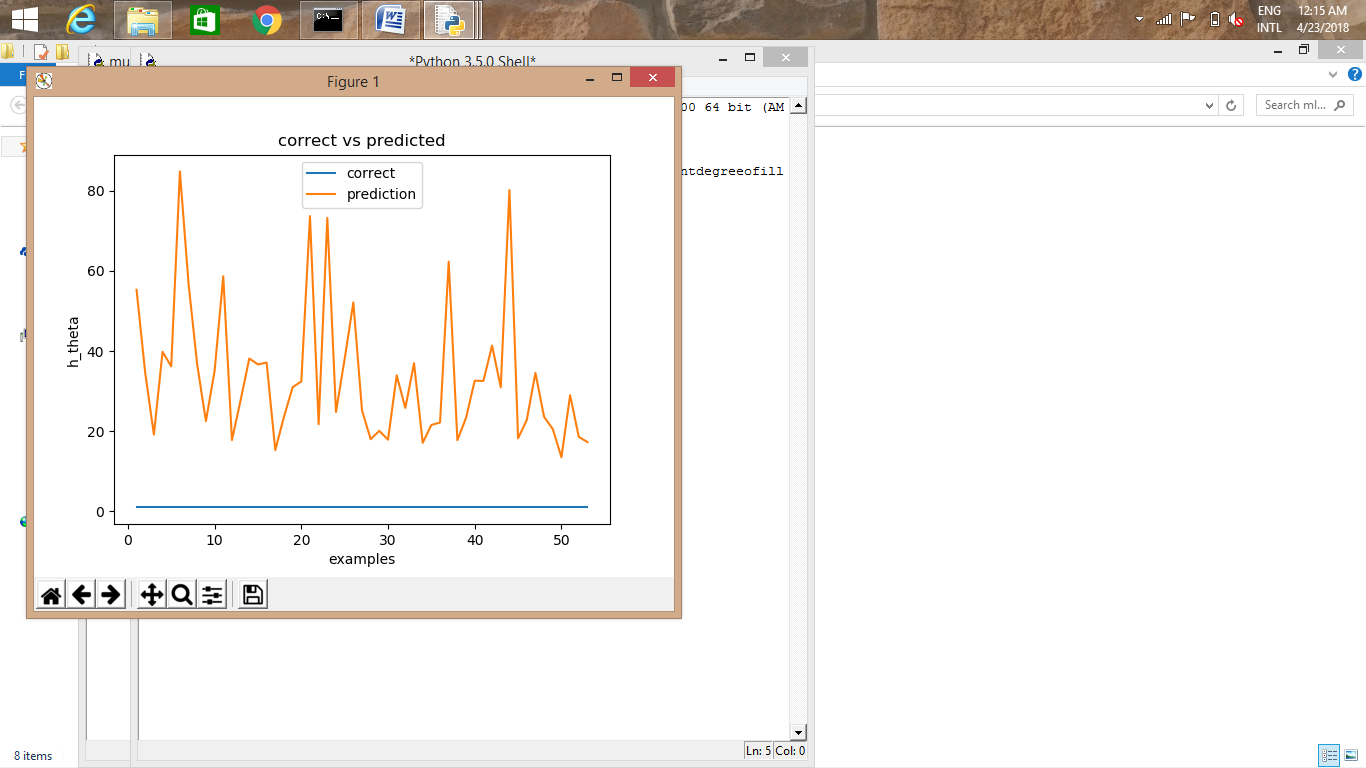
plt.legend()

# function to show the plot

plt.show()

OUTPUT:





3.OCTANE

import numpy as np

import csv

import matplotlib.pyplot as plt

n = 5 # number of input features.

m = 82 # number of training examples.

grad = np.zeros(shape = (n, 1))

theta = np.ones(shape=(n, 1), dtype = float)

hx = np.ones(shape=(m, 1), dtype = float)

file\_handle = open("datasets/octane/data.csv", "r")

reader = csv.reader(file\_handle, delimiter = ',')

learning\_rate = 1e-6

def h(X):

global theta

res = np.matmul(np.transpose(theta), X)

return res

cost\_list = []

itr\_list = []

def gradient\_descent\_algorithm():

global theta, grad

num\_itrs = 100

for itr in range(num\_itrs):

file\_handle.seek(0)

total\_cost = 0.0

idx = 0

for row in reader:

X = [float(x) for x in row[0: -1]]

# list of floats

X = np.asarray(X)

np.reshape(X, [n, 1])

hx[idx][0] = h(X)

y\_correct = float(row[0])

diff = (hx[idx][0] - y\_correct)

total\_cost += (diff \* diff)

idx += 1

for j in range(n):

grad[j][0] = 0.0

i = 0

file\_handle.seek(0)

for row in reader:

y\_correct = float(row[-1])

xij = float(row[j + 1])

diff = hx[i][0] - y\_correct

grad[j][0] += ((learning\_rate \* diff \* xij) / m)

i += 1

theta = theta - grad

total\_cost = total\_cost /(2 \* m)

cost\_list.append(total\_cost)

itr\_list.append(itr + 1)

gradient\_descent\_algorithm()

plt.plot(itr\_list, cost\_list, label = "cost")

plt.xlabel("iterations")

# naming the y axis

plt.ylabel('Cost')

# giving a title to my graph

plt.title('Cost vs iterations')

# show a legend on the plot

plt.legend()

# function to show the plot

plt.show()

ypaxis = []

ycaxis = []

xaxis = []

index = 0

file\_handle.seek(0)

for row in reader:

X = [float(x) for x in row[1:]]

# list of floats

X = np.asarray(X)

np.reshape(X, [n, 1])

pred = h(X)

y\_correct = float(row[0])

index += 1

ypaxis.append(pred)

ycaxis.append(y\_correct)

xaxis.append(index)

plt.plot(xaxis, ycaxis, label = "correct")

plt.plot(xaxis, ypaxis, label = "prediction")

plt.xlabel("examples")

# naming the y axis

plt.ylabel('h\_theta')

plt.title('correct vs predicted')

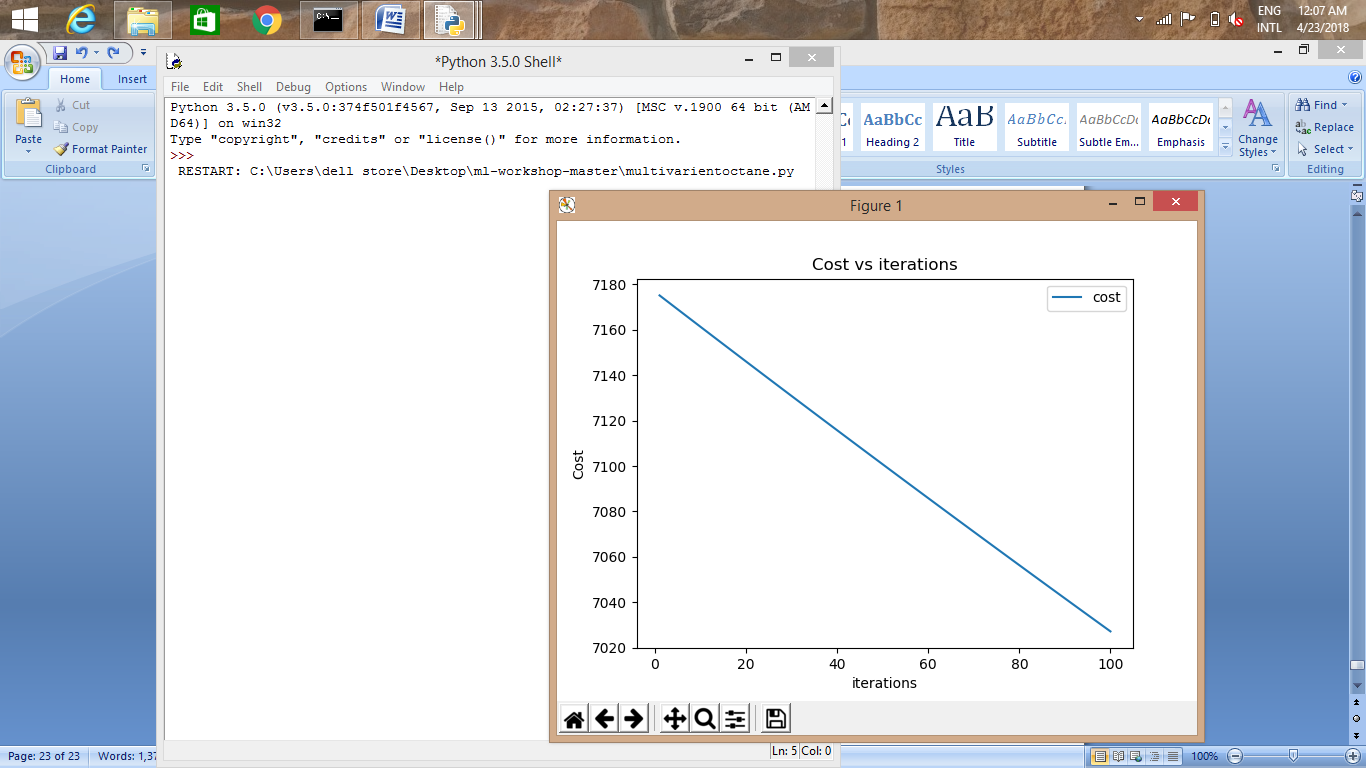
# show a legend on the plot

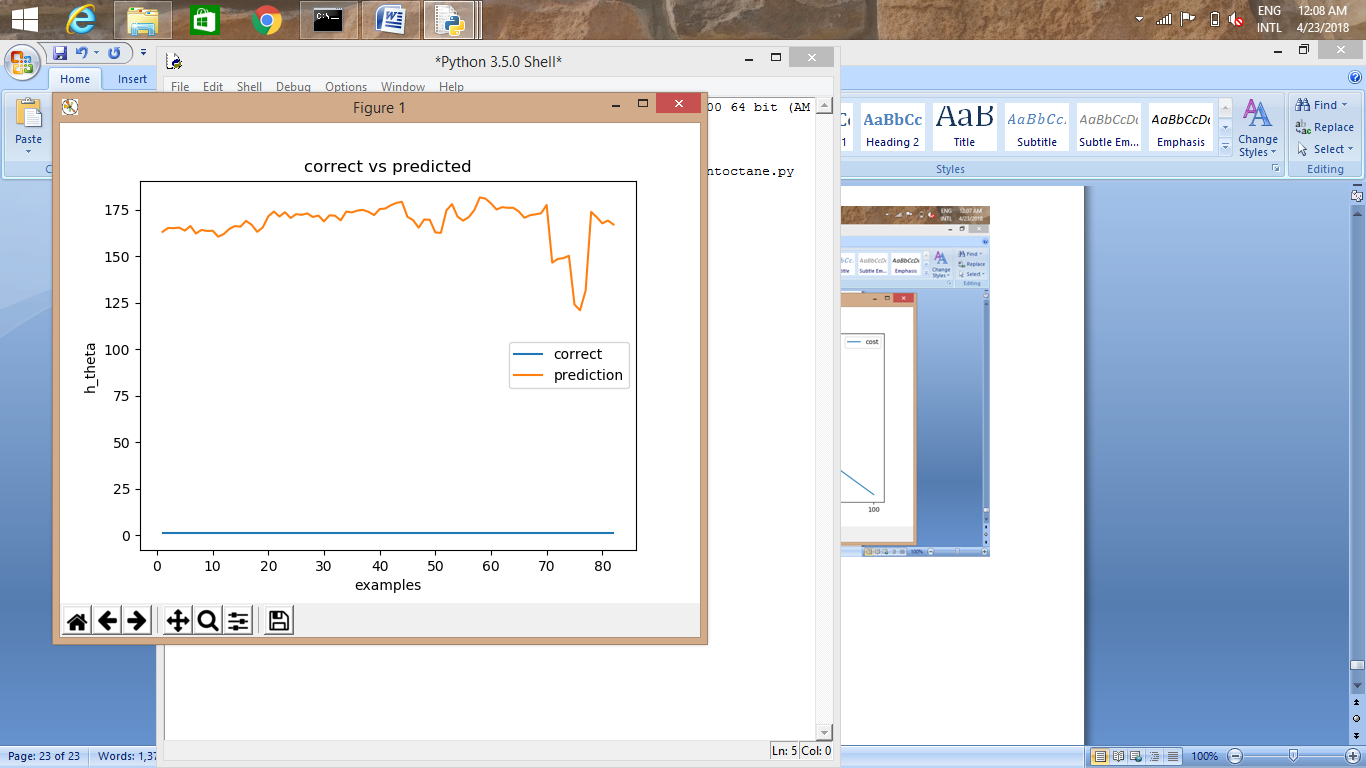
plt.legend()

# function to show the plot

plt.show()

OUTPUT:





4.PASTURE-RENT-STRUCTURE

import numpy as np

import csv

import matplotlib.pyplot as plt

n = 4 # number of input features.

m = 67 # number of training examples.

grad = np.zeros(shape = (n, 1))

theta = np.ones(shape=(n, 1), dtype = float)

hx = np.ones(shape=(m, 1), dtype = float)

file\_handle = open("datasets/pasture-rent-structure/data.csv", "r")

reader = csv.reader(file\_handle, delimiter = ',')

learning\_rate = 1e-6

def h(X):

global theta

res = np.matmul(np.transpose(theta), X)

return res

cost\_list = []

itr\_list = []

def gradient\_descent\_algorithm():

global theta, grad

num\_itrs = 100

for itr in range(num\_itrs):

file\_handle.seek(0)

total\_cost = 0.0

idx = 0

for row in reader:

X = [float(x) for x in row[0: -1]]

# list of floats

X = np.asarray(X)

np.reshape(X, [n, 1])

hx[idx][0] = h(X)

y\_correct = float(row[0])

diff = (hx[idx][0] - y\_correct)

total\_cost += (diff \* diff)

idx += 1

for j in range(n):

grad[j][0] = 0.0

i = 0

file\_handle.seek(0)

for row in reader:

y\_correct = float(row[-1])

xij = float(row[j + 1])

diff = hx[i][0] - y\_correct

grad[j][0] += ((learning\_rate \* diff \* xij) / m)

i += 1

theta = theta - grad

total\_cost = total\_cost /(2 \* m)

cost\_list.append(total\_cost)

itr\_list.append(itr + 1)

gradient\_descent\_algorithm()

plt.plot(itr\_list, cost\_list, label = "cost")

plt.xlabel("iterations")

# naming the y axis

plt.ylabel('Cost')

# giving a title to my graph

plt.title('Cost vs iterations')

# show a legend on the plot

plt.legend()

# function to show the plot

plt.show()

ypaxis = []

ycaxis = []

xaxis = []

index = 0

file\_handle.seek(0)

for row in reader:

X = [float(x) for x in row[1:]]

# list of floats

X = np.asarray(X)

np.reshape(X, [n, 1])

pred = h(X)

y\_correct = float(row[0])

index += 1

ypaxis.append(pred)

ycaxis.append(y\_correct)

xaxis.append(index)

plt.plot(xaxis, ycaxis, label = "correct")

plt.plot(xaxis, ypaxis, label = "prediction")

plt.xlabel("examples")

# naming the y axis

plt.ylabel('h\_theta')

plt.title('correct vs predicted')

# show a legend on the plot

plt.legend()

# function to show the plot

plt.show()

OUTPUT:

