**Code:**

"""

Data Mining\_Assignment 3

Name: Kaiqin Huang

Date Due: Feb 16, 2017

Consider the attached csv file, which lists high school and college GPAs

for 105 students.

Write a python module that computes a linear regression model to learn

the relationship between high school and university GPA.

In addition to learning the parameters for the best fit line,

plot the data points and line using PyPlot (library comes with anaconda).

You must write your own code to compute the regression parameters

and may not use a built-in library function.

Submit a word doc with your plot and python code in a single file.

"""

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import math

def corr(list1, list2):

"""Calculate the correlation of two lists

Args:

list1 and list2: list of floats

Return:

cr: correlation

"""

len\_list = len(list1)

if len(list2) != len\_list:

print("Two lists have different lengths.")

else:

aver\_x = aver(list1)

aver\_y = aver(list2)

sum\_xy = 0

sum\_xx = 0

sum\_yy = 0

for i in range(len\_list):

dev\_x = list1[i] - aver\_x

dev\_y = list2[i] - aver\_y

sum\_xy += dev\_x \* dev\_y

sum\_xx += dev\_x \* dev\_x

sum\_yy += dev\_y \* dev\_y

cr = sum\_xy / math.sqrt(sum\_xx \* sum\_yy)

return cr

def aver(list\_input):

"""Calculate the mean of numbers in a list

Args:

list: a list of floats

Return:

aver\_list: mean

"""

len\_list = len(list\_input)

sum\_list = 0

for i in range(len\_list):

sum\_list += list\_input[i]

aver\_list = sum\_list / len\_list

return aver\_list

def sd(list\_input):

"""Calculate the standard deviation of numbers in a list

Args:

list: a list of floats

Return:

sd\_list: standard deviation

"""

aver\_list = aver(list\_input)

len\_list = len(list\_input)

sum\_dev = 0

for i in range(len\_list):

sum\_dev += (list\_input[i] - aver\_list) \* (list\_input[i] - aver\_list)

sd\_list = math.sqrt(sum\_dev / (len\_list - 1))

return sd\_list

def test\_corr():

assert corr([1,2,3], [1,2,3]) == 1

def test\_aver():

assert aver([1,2,3]) == 1

def test\_sd():

assert sd([1,2,3]) == 1

df = pd.read\_csv("GPA\_Reg.csv", header = 0, nrows = 105)

high\_GPA = df["high\_GPA"]

univ\_GPA = df["univ\_GPA"]

m\_est = corr(high\_GPA, univ\_GPA) \* sd(univ\_GPA) / sd(high\_GPA)

b\_est = aver(univ\_GPA) - m\_est \* aver(high\_GPA)

print("The estimated values of m and b are:")

print(m\_est, b\_est)

print("Using built-in regression function in Numpy, the estimated m and b are:")

print(np.polyfit(high\_GPA, univ\_GPA, 1))

univ\_GPA\_est = []

for i in range(len(high\_GPA)):

univ\_GPA\_predict = m\_est \* high\_GPA[i] + b\_est

univ\_GPA\_est.append(univ\_GPA\_predict)

plt.scatter(high\_GPA, univ\_GPA)

plt.plot(high\_GPA, univ\_GPA\_est)

plt.savefig("Assignment3\_GPA\_regression.png")

**Output:**

runfile('/Users/kaiqinhuang/Desktop/Assignment3\_KaiqinHuang.py', wdir='/Users/kaiqinhuang/Desktop')

The estimated values of m and b are:

0.674829903448 1.09682328179

Using built-in regression function in Numpy, the estimated m and b are:

[ 0.6748299 1.09682328]

