HOMEWORK 1

Q1.

import numpy as np

from numpy import genfromtxt

from numpy import linalg as lg

from numpy import linalg as LA

from matplotlib import pyplot as plt

import pandas as pd

def d\_PCA(x):

columnMean = x.mean(axis=0)

columnMeanAll = np.tile(columnMean, reps=(x.shape[0], 1))

xMeanCentered = x - columnMeanAll

#use mean\_centered data or standardized mean\_centered data

dataForPca = xMeanCentered

#get covariance matrix of the data

covarianceMatrix = np.cov(dataForPca, rowvar=False)

#eigendecomposition of the covariance matrix

eigenValues, eigenVectors = LA.eig(covarianceMatrix)

II = eigenValues.argsort()[::-1]

eigenValues = eigenValues[II]

eigenVectors = eigenVectors[:, II]

#get scores

pcaScores = np.matmul(dataForPca, eigenVectors)

#collect PCA results

pcaResults = {'data': x,

'mean\_centered\_data': xMeanCentered,

'PC\_variance': eigenValues,'loadings': eigenVectors,

'scores': pcaScores}

return pcaResults

data\_1 = genfromtxt("C:/Users/vyoms/Desktop/linear\_regression\_test\_data.csv", delimiter=",")

data = data\_1[1:,1:3]

data\_x = data\_1[1:,1]

data\_y = data\_1[1:,2]

data\_yth = data\_1[1:,3]

myPCAResults = d\_PCA(data)

X = data\_x

Y = data\_y

# Total no. of X values

m = len(X)

# Mean of X and Y values

mean\_x = np.mean(X)

mean\_y = np.mean(Y)

# calculating b1 and b2

numerator = 0

denominator = 0

for i in range(m):

numerator += (X[i] - mean\_x) \* (Y[i] - mean\_y)

denominator += (X[i] - mean\_x) \*\* 2

b1 = numerator / denominator

b0 = mean\_y - (b1 \* mean\_x)

max\_x = np.max(X)

min\_x = np.min(X)

# Calculating line values x and y

x = np.linspace(min\_x, max\_x, 1000)

y = b0 + b1 \* x

fig = plt.figure()

ax = fig.add\_subplot(1,1,1)

ax.set\_title('scores plot')

ax.scatter(data\_x, data\_y, color='tomato')

ax.scatter(data\_x,data\_yth, color='dodgerblue')

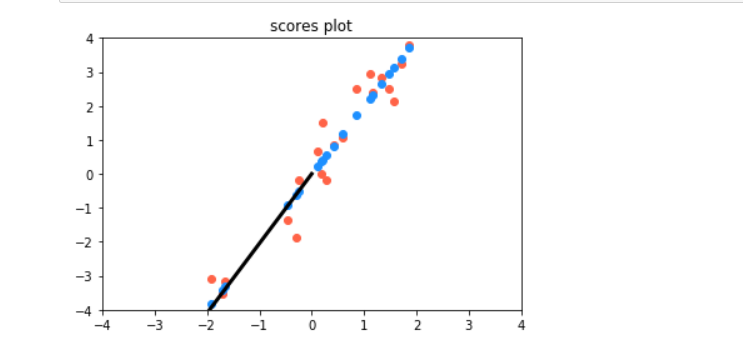
ax.plot([0,200\*myPCAResults['loadings'][0,0]], [0, 200\*myPCAResults['loadings'][1,0]],

color='black', linewidth=3)

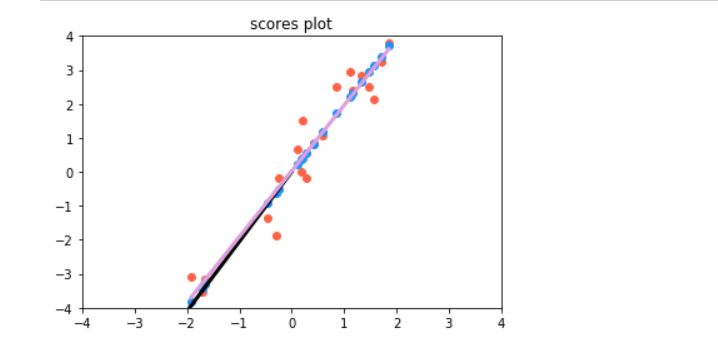
plt.plot(x, y, color='plum', linewidth=3, label='Regression Line')

plt.xlim(-4, 4), plt.ylim(-4, 4)

plt.show()

a.   


b.



In the above graph the black line is the PC1 line and plum colored is the Regression line. It is clearly visible from the way that the lines are overlapping which means they are similar.

Q2.

import matplotlib.pyplot as plt

import random

import numpy as np

from sklearn import datasets, linear\_model

from sklearn.utils import shuffle

from sklearn.metrics import mean\_squared\_error, r2\_score

# Load the diabetes dataset

diabetes = datasets.load\_diabetes()

shuffle(diabetes['data'])

shuffle(diabetes['target'])

# We ll be using only 1 feature out of 10

X = diabetes.data[:, np.newaxis, 2]

type(X)

X\_test = X[:20]

X\_train = X[20:]

X\_train

y\_test = diabetes.target[:20]

y\_train = diabetes.target[20:]

regr = linear\_model.LinearRegression()

regr.fit(X\_train, y\_train)

# Make predictions using the testing set

y\_pred = regr.predict(X\_test)

# Plot outputs

plt.scatter(X\_test, y\_test, color='red')

plt.plot(X\_test, y\_pred, color='black')

plt.show()

