**Muhammad Abdullah Zainab Ijaz**

**19F-0916 19F-0917**

Applied Machine Learning

Assignment # 1

**Question # 1**

**Part (a)**

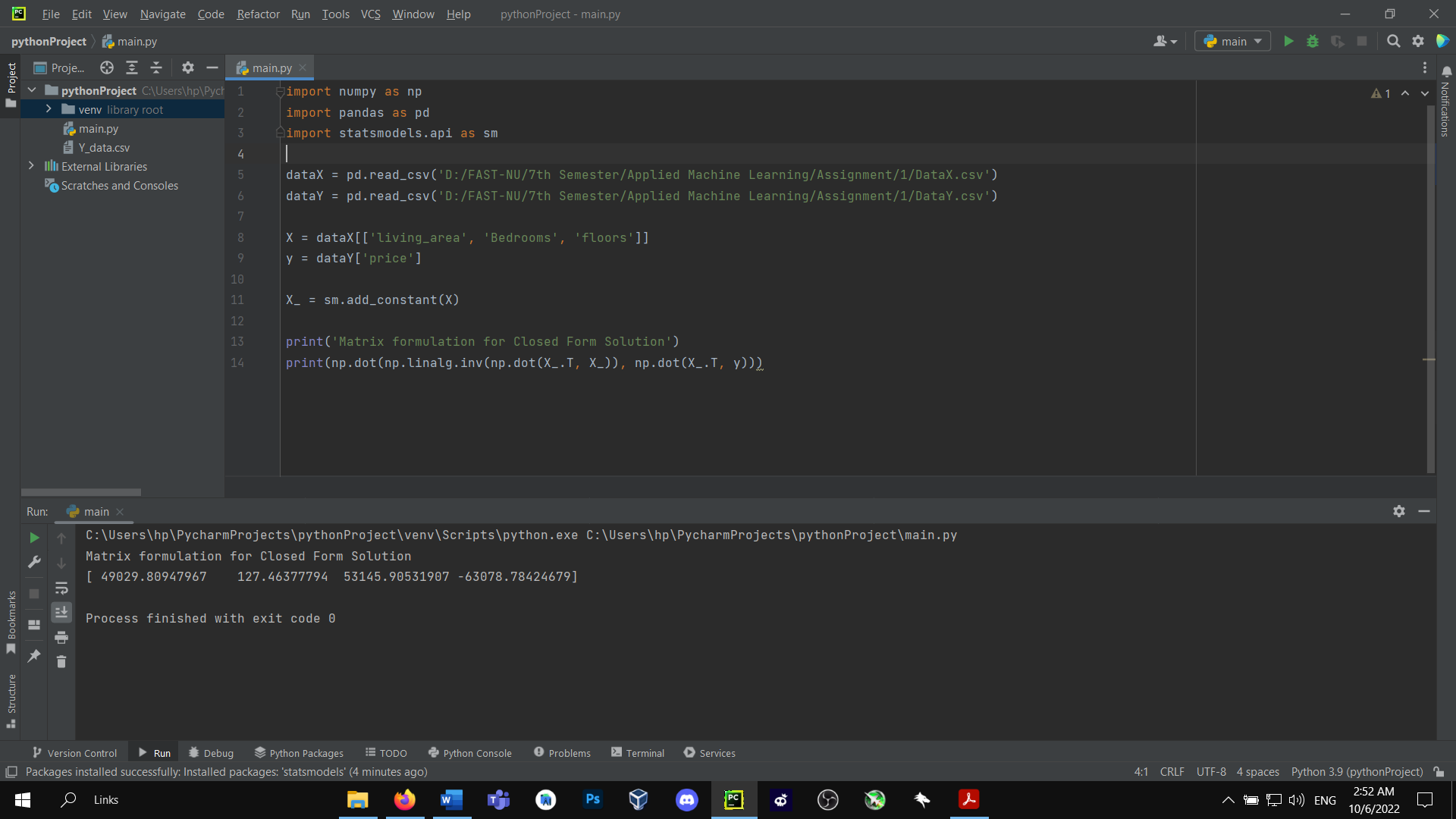
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
def cost\_function(X, y, theta):  
 m = y.size  
 error = np.dot(X, theta.T) - y  
 cost = 1/(2\*m) \* np.dot(error.T, error)  
 return cost, error  
  
def gradient\_descent(X, y, theta, alpha, iterations):  
 cost\_array = np.zeros(iterations)  
 m = y.size  
 for i in range(iterations):  
 cost, error = cost\_function(X, y, theta)  
 theta = theta - (alpha \* (1/m) \* np.dot(X.T, error))  
 cost\_array[i] = cost  
 return theta, cost\_array  
  
def plotChart(iterations, cost\_num):  
 fig, ax = plt.subplots()  
 ax.plot(np.arange(iterations), cost\_num, 'r')  
 ax.set\_xlabel('Iterations')  
 ax.set\_ylabel('Error')  
 ax.set\_title('Error vs Iterations')  
 plt.show()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 dataX = pd.read\_csv('D:/FAST-NU/7th Semester/Applied Machine Learning/Assignment/1/DataX.csv')  
 dataY = pd.read\_csv('D:/FAST-NU/7th Semester/Applied Machine Learning/Assignment/1/DataY.csv')  
  
 X = dataX[['living\_area', 'Bedrooms', 'floors']]  
 y = dataY['price']  
  
 X = (X - X.mean()) / X.std()  
 X = np.c\_[np.ones(X.shape[0]), X]  
  
 alpha = 0.02  
 iterations = 1000  
  
 theta = np.ones(X.shape[1])  
 initial\_cost, \_ = cost\_function(X, y, theta)  
  
 print('With initial theta values of {1}, cost error is {1}'.format(theta, initial\_cost))  
  
 theta, cost\_num = gradient\_descent(X, y, theta, alpha, iterations)  
  
 plotChart(iterations, cost\_num)  
  
 final\_cost, \_ = cost\_function(X, y, theta)  
  
 print('With final theta values of {0}, cost error is {1}'.format(theta, final\_cost))

**Text

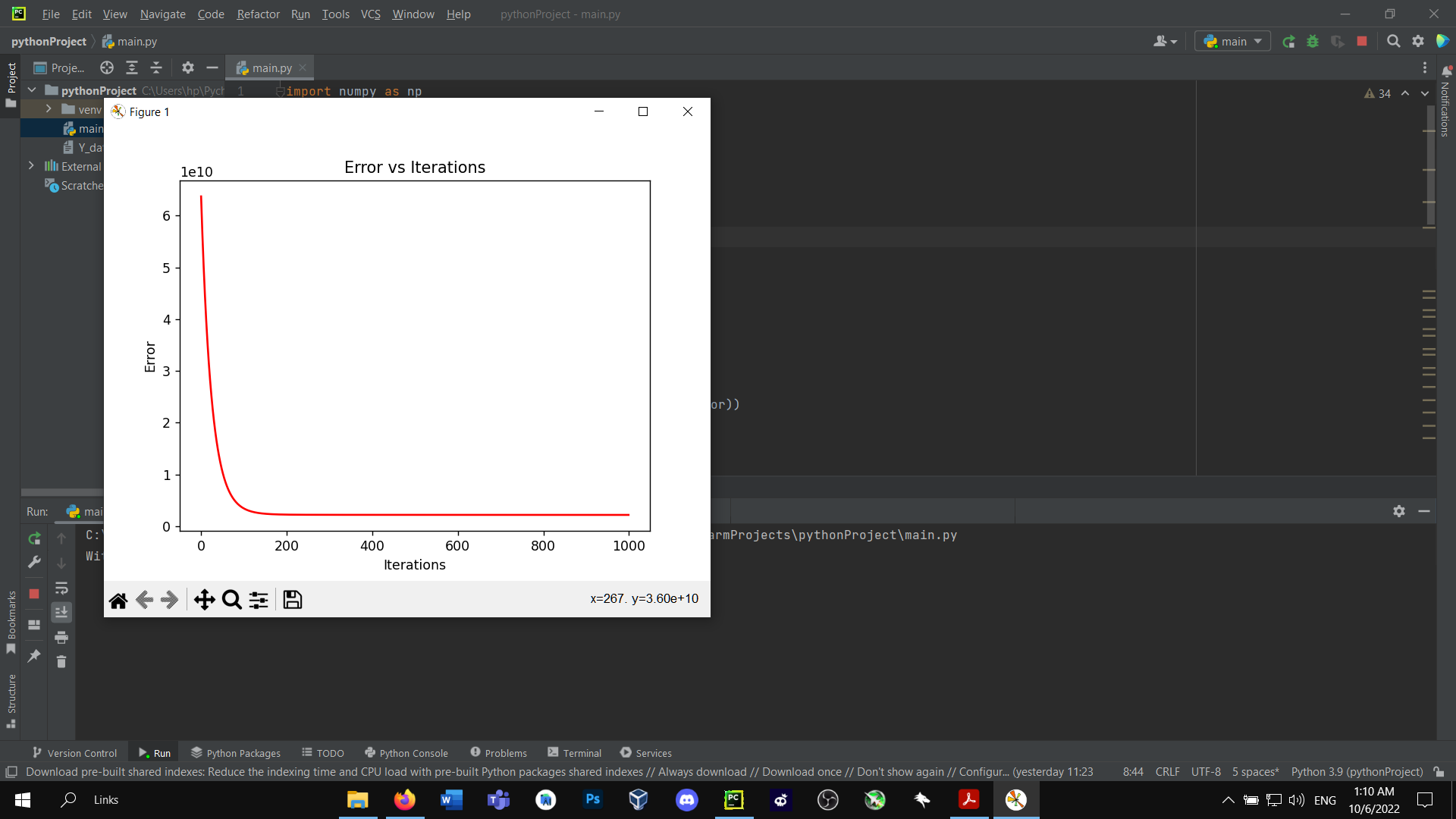
Description automatically generated**

**Part (b)**

import numpy as np  
import pandas as pd  
import statsmodels.api as sm  
  
dataX = pd.read\_csv('D:/FAST-NU/7th Semester/Applied Machine Learning/Assignment/1/DataX.csv')  
dataY = pd.read\_csv('D:/FAST-NU/7th Semester/Applied Machine Learning/Assignment/1/DataY.csv')  
  
X = dataX[['living\_area', 'Bedrooms', 'floors']]  
y = dataY['price']  
  
X\_ = sm.add\_constant(X)  
  
print('Matrix formulation for Closed Form Solution')  
print(np.dot(np.linalg.inv(np.dot(X\_.T, X\_)), np.dot(X\_.T, y)))

****

**Part (c)**

****

**Part (d)**

There is too much difference in part(a) and part(b) because as we can see in the graph, the line is not a straight line but is something like a trajectory. We know that the Closed-Form Solution works for only straight-line solutions which is why the difference between Gradient Decent by Regression modeling is way much different from closed-form solutions.

The solution in part (a) very much depicts the original solution for the given problem

**Question # 3**

**Logistic Regressions**

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
  
  
def sigmoid(z):  
 return 1 / (1 + np.exp(-z))  
  
  
def cost(h, y):  
 return (-y \* np.log(h) - (1 - y) \* np.log(1 - h)).mean()  
  
  
def gradient(X, h, y):  
 return np.dot(X.T, (h - y)) / y.shape[0]  
  
  
def logistic\_regression(X, y, theta, alpha, iterations):  
 cost\_array = np.zeros(iterations)  
 for i in range(iterations):  
 h = sigmoid(np.dot(X, theta))  
 cost\_num = cost(h, y)  
 cost\_array[i] = cost\_num  
 gradient\_val = gradient(X, h, y)  
 theta = theta - (gradient\_val \* alpha)  
 return theta, cost\_array  
  
  
def plotChart(iterations, cost\_num):  
 fig, ax = plt.subplots()  
 ax.plot(np.arange(iterations), cost\_num, 'r')  
 ax.set\_xlabel('Iterations')  
 ax.set\_ylabel('Error')  
 ax.set\_title('Error vs Iterations')  
 plt.show()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 dataX = pd.read\_csv('D:/FAST-NU/7th Semester/Applied Machine Learning/Assignment/1/DataX.csv')  
 dataY = pd.read\_csv('D:/FAST-NU/7th Semester/Applied Machine Learning/Assignment/1/ClassY.csv')  
  
 X = dataX[['living\_area', 'Bedrooms', 'floors']]  
 y = dataY['Type']  
  
 X = np.concatenate((np.ones((X.shape[0], 1)), X), axis=1)  
  
 theta = np.zeros(X.shape[1])  
  
 alpha = 0.02  
 iterations = 1000  
  
 h = sigmoid(np.dot(X, theta))  
 print("Initial cost value for theta values {0} is: {1}".format(theta, cost(h, y)))  
  
 theta, cost\_num = logistic\_regression(X, y, theta, alpha, iterations)  
  
 plotChart(iterations, cost\_num)  
  
 h = sigmoid(np.dot(X, theta))  
 print("Final cost value for theta values {0} is: {1}".format(theta, cost(h, y)))

