Project

July 4, 2023

[ ]:
```
import numpy as np
from matplotlib import pyplot as plt
```

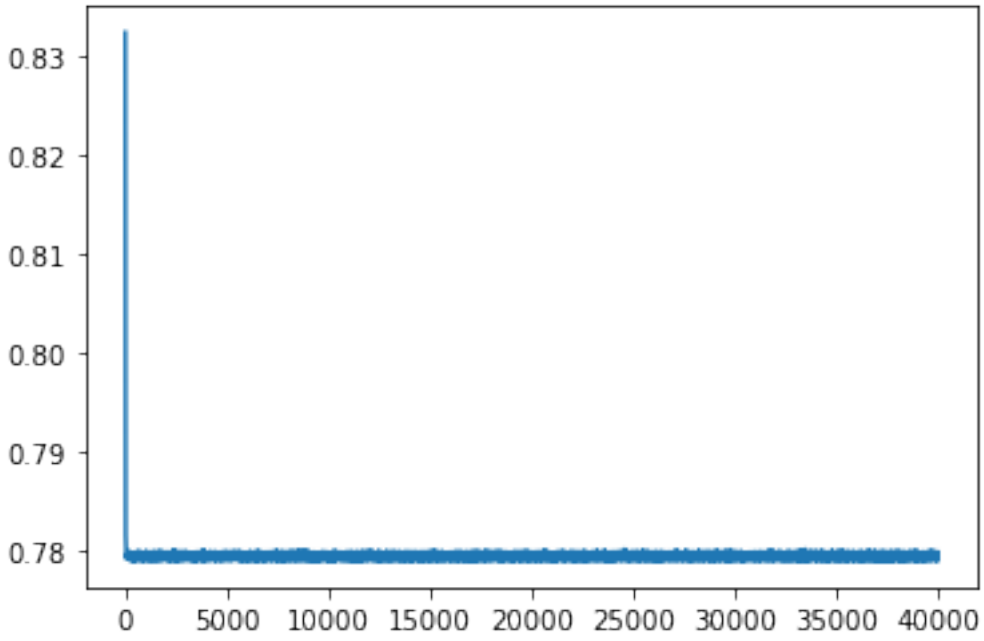
0.1 Q1)

[ ]:
```
n, m = 10, 100
np.random.seed(1)
Z = np.random.normal(0,1,(m,n))
B = np.random.normal(0, 1, m)
```

[ ]:
```
def gradiantDescent(x0, eta):
    x = x0
    T = 40000
    cost = []
    for i in range(T):
        loss = np.abs(Z@x - B).sum()/m
        gradiant = np.zeros(n)
        for j in range(m):
            if Z[j,:]@x-B[j] >0:
                gradiant += Z[j,:]
            else:
                gradiant -= Z[j,:]
        gradiant /= m
        x -= eta*gradiant
        cost.append(loss)

    return cost
```

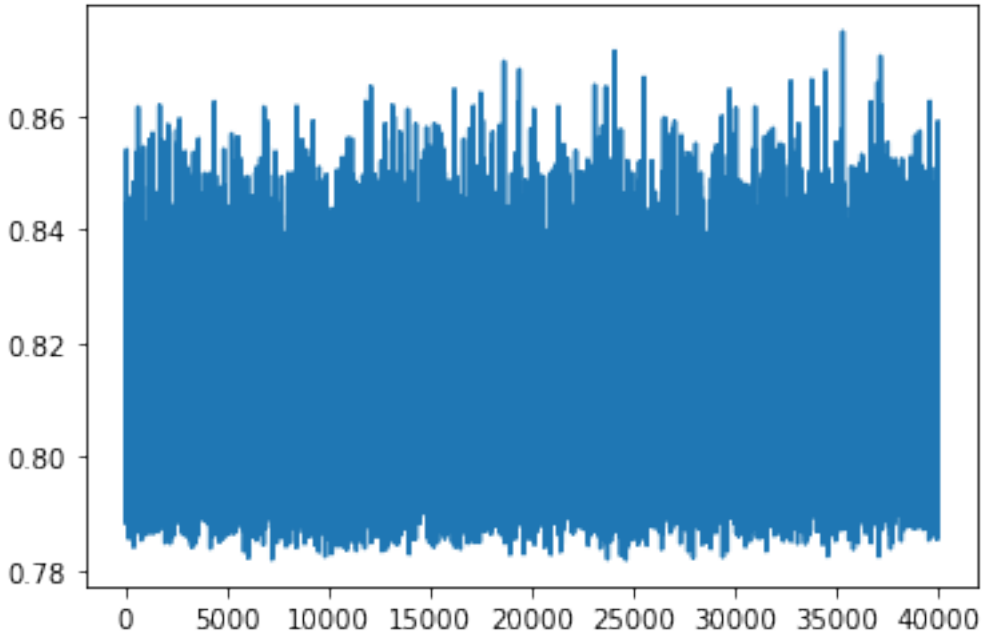
[ ]:
```
cost1 = gradiantDescent(np.zeros(n), .1)
plt.plot(range(40000), cost1)
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x107d45430>]
```



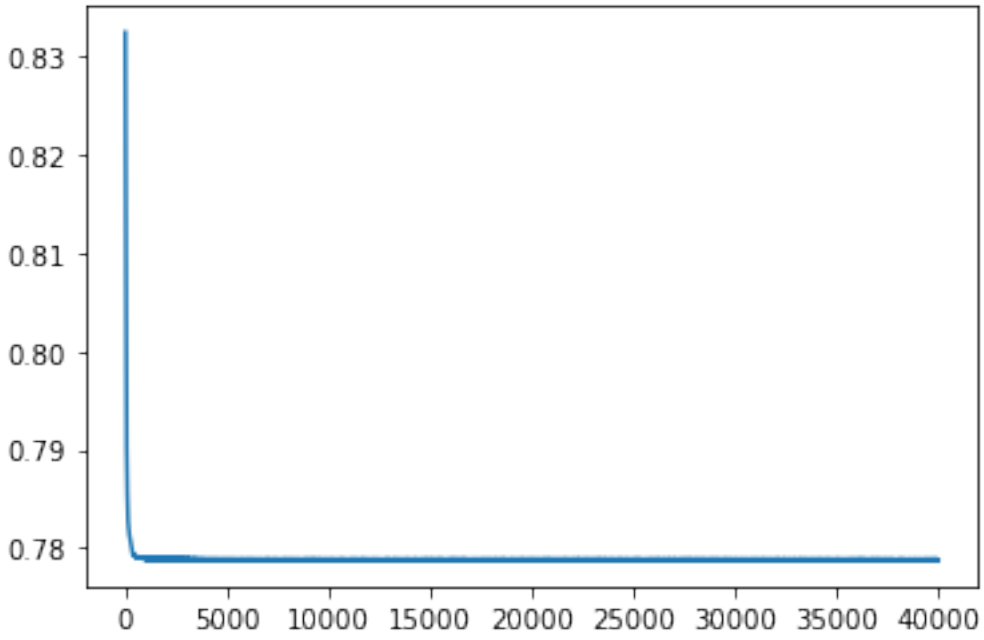
[ ]:
```
cost2 = gradiantDescent(np.zeros(n), 1)
plt.plot(range(40000), cost2)
```

[ ]: [<matplotlib.lines.Line2D at 0x107f81700>]



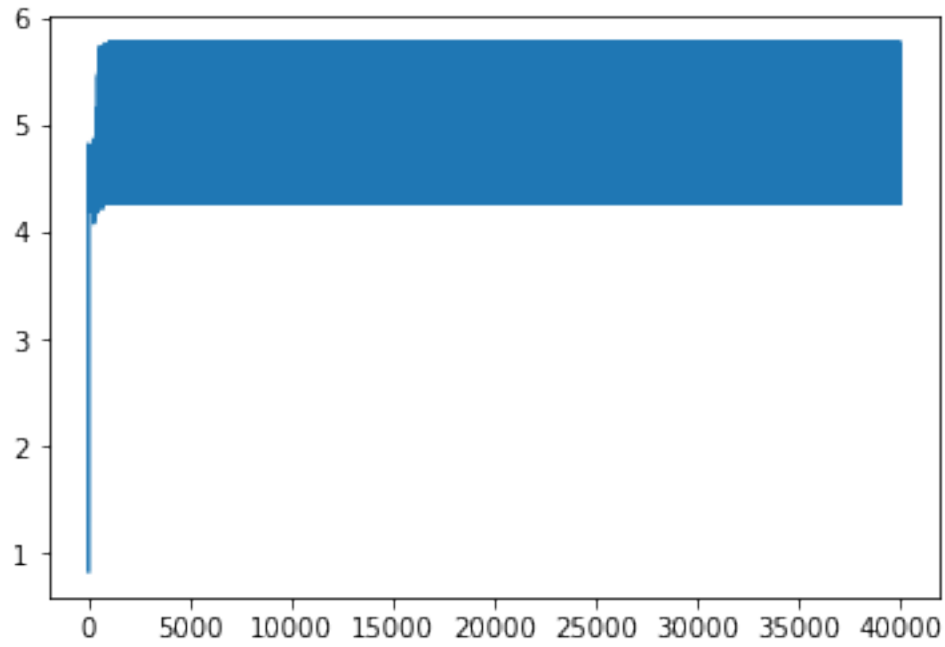
[ ]:
```
cost3 = gradiantDescent(np.zeros(n), .01)
plt.plot(range(40000), cost3)
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x111d5a7c0>]
```



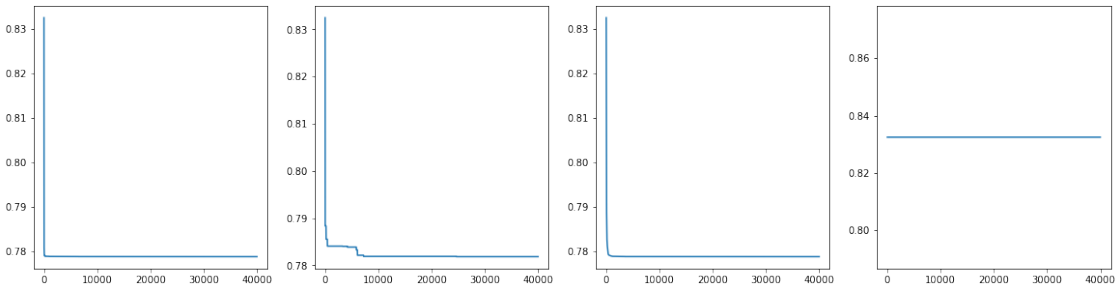
[ ]:
```
cost4 = gradiantDescent(np.zeros(n), 10)
plt.plot(range(40000), cost4)
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x11681c9a0>]
```



0.2 Q2)

[ ]:
```
f, a = plt.subplots(1, 4, figsize=(20,5))
costs = [cost1, cost2, cost3, cost4]
for e in range(4):
    minls = []
    for i in range(len(costs[e])):
        minls.append(min(costs[e][:i+1]))

    a[e].plot(range(40000), minls)
```

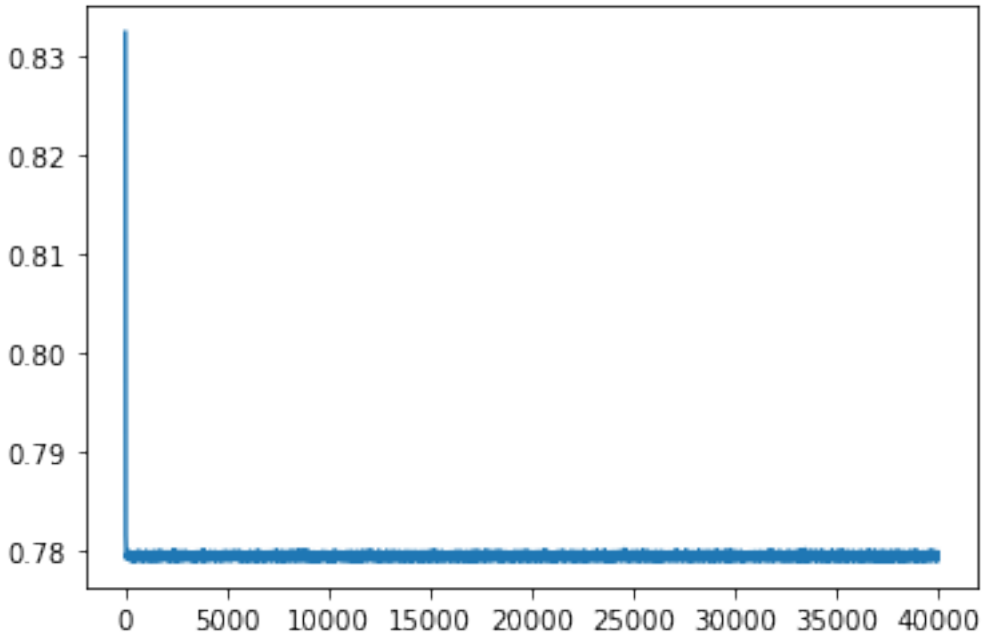


1 Q3)

[ ]:
```
def stochasticGradiantDescent(x0, eta):
    x = x0
    T = 40000
    cost = []
    for i in range(T):
        loss = np.abs(Z@x - B).sum()/m
        gradiant = np.zeros(n)
        for j in range(m):
            if Z[j,:]@x-B[j] >0:
                gradiant += Z[j,:]
            else:
                gradiant -= Z[j,:]
        gradiant /= m
        gradiant += (np.random.rand(n)*2-1) * np.linalg.norm(gradiant) * .1
        x -= eta*gradiant
        cost.append(loss)

    return cost
```

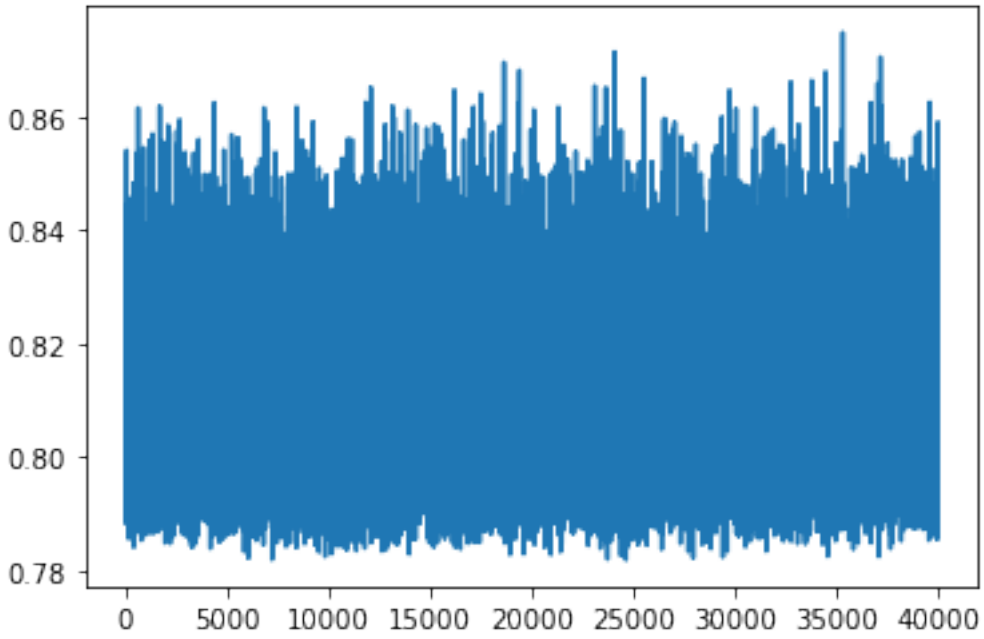
[ ]:
```
st_cost1 = stochasticGradiantDescent(np.zeros(n), .01)
plt.plot(range(40000), cost1)
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x116b3ba90>]
```



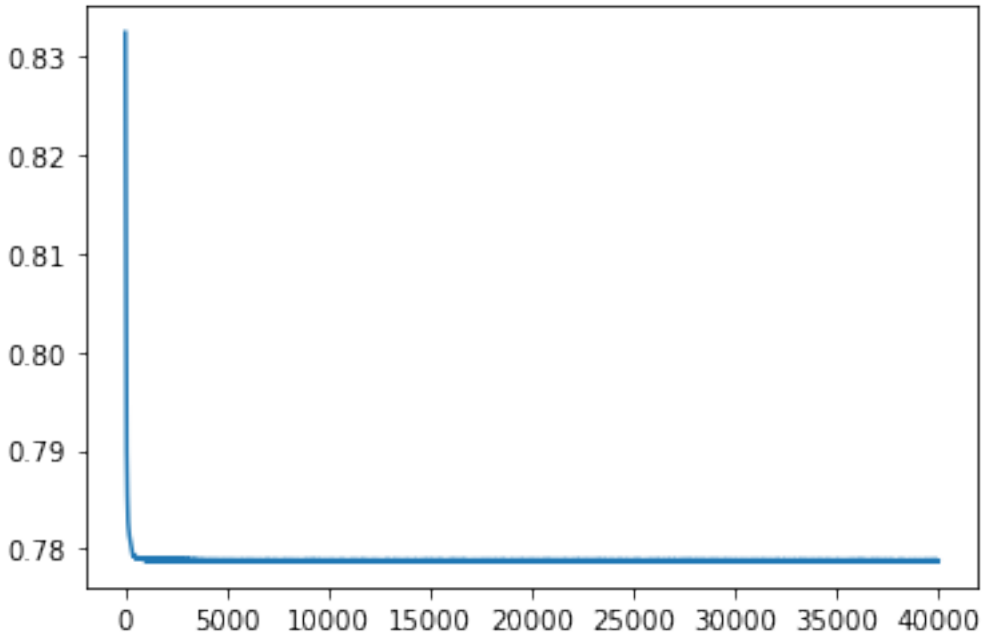
[ ]:
```
st_cost2 = stochasticGradiantDescent(np.zeros(n), .1)
plt.plot(range(40000), cost2)
```

[ ]: [<matplotlib.lines.Line2D at 0x116ce2970>]



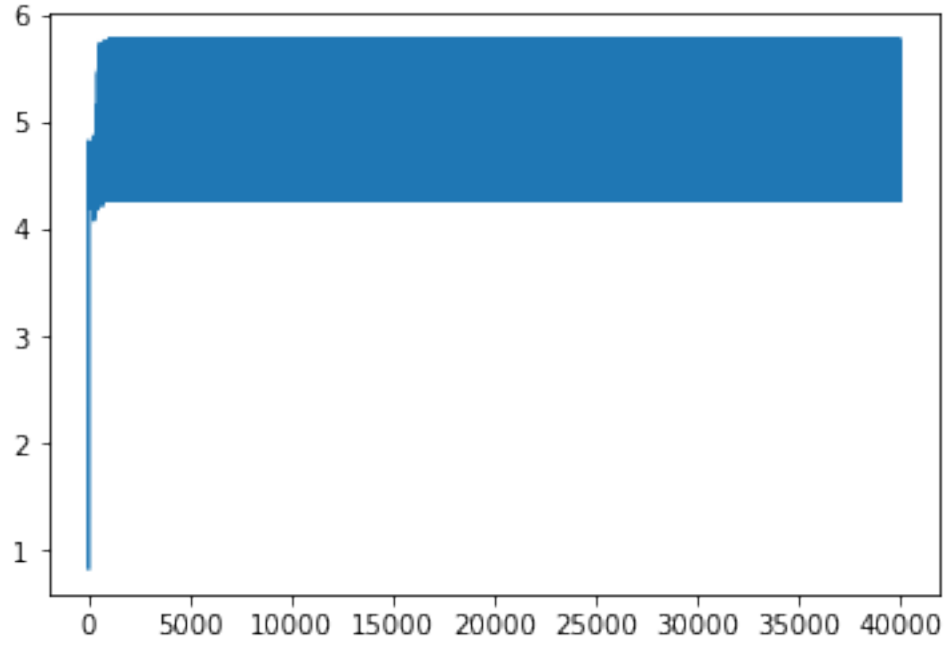
[ ]:
```
st_cost3 = stochasticGradiantDescent(np.zeros(n), 1)
plt.plot(range(40000), cost3)
```

[ ]: [<matplotlib.lines.Line2D at 0x1176baa30>]



[ ]:
```
st_cost4 = stochasticGradiantDescent(np.zeros(n), 10)
plt.plot(range(40000), cost4)
```

[ ]: [<matplotlib.lines.Line2D at 0x117865760>]

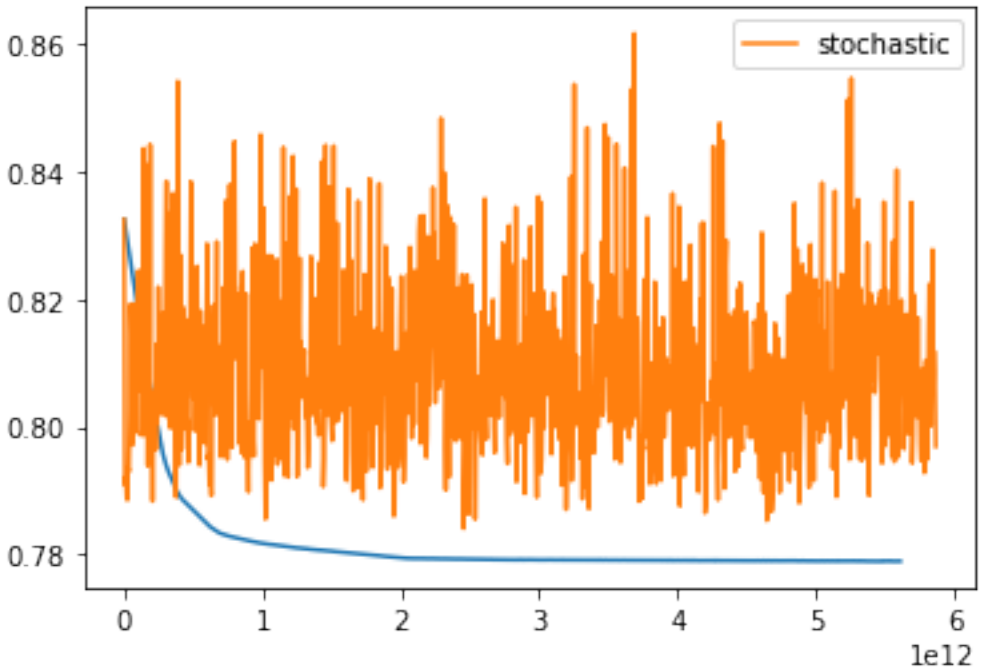


1.1 Q4)

[ ]:
```
import time
```

[ ]:
```
t1 = time.time_ns()
cost1 = gradiantDescent(np.zeros(n), .01)
t1 = time.time_ns() - t1

t2 = time.time_ns()
st_cost1 = stochasticGradiantDescent(np.zeros(n), .01)
t2 = time.time_ns() - t2

plt.plot(np.arange(1000) * t1, cost1[:1000])
plt.plot(np.arange(1000) * t2, cost2[:1000], label='stochastic')
plt.legend()
```

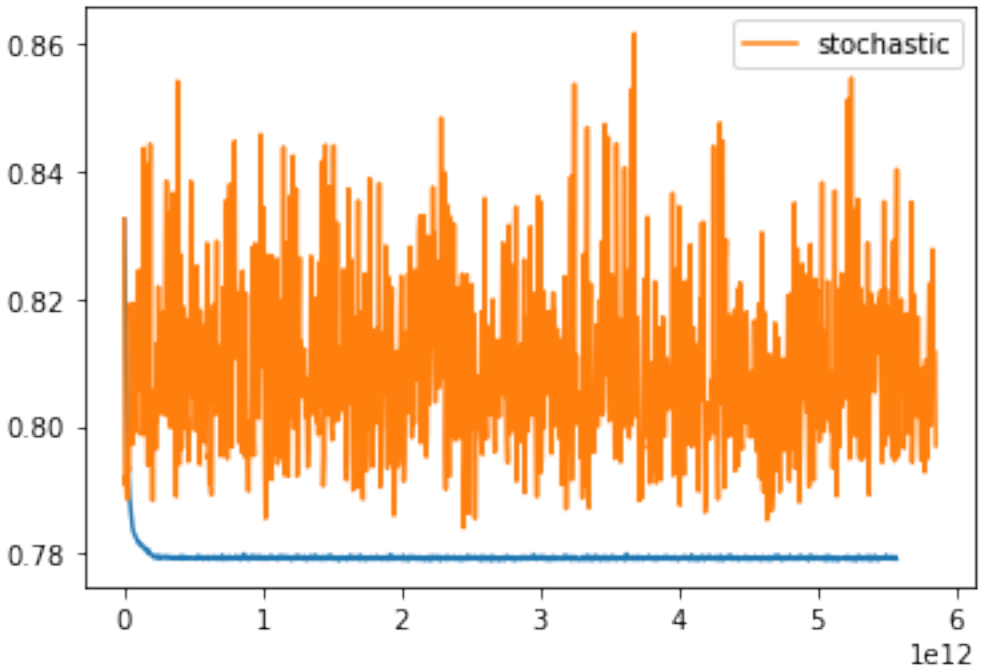
[ ]:
```
<matplotlib.legend.Legend at 0x1169a12e0>
```



[ ]:
```
t1 = time.time_ns()
cost1 = gradiantDescent(np.zeros(n), .1)
t1 = time.time_ns() - t1

t2 = time.time_ns()
```

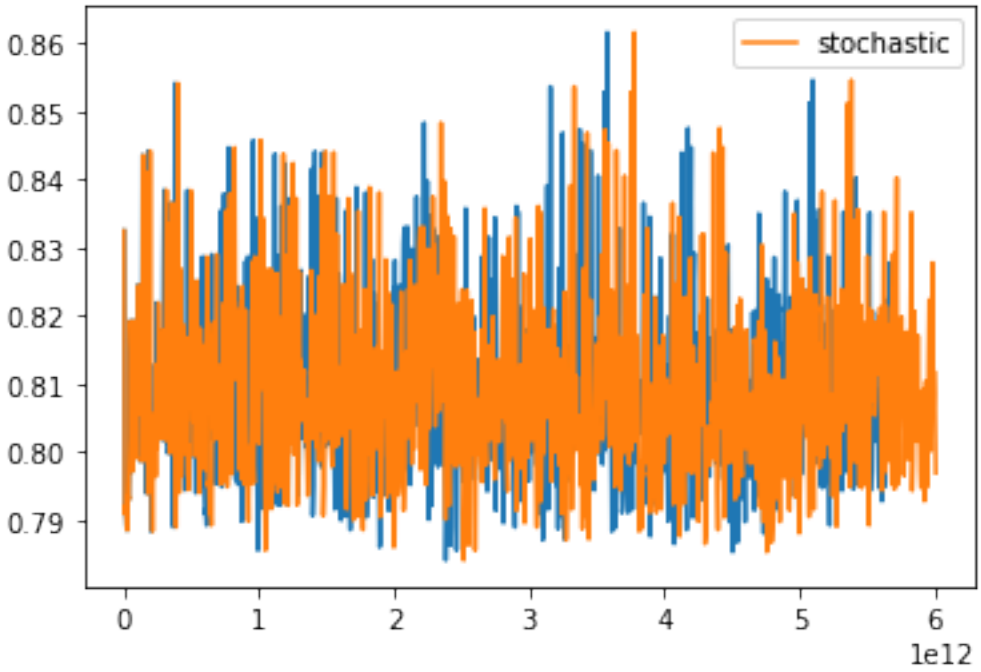
```
st_cost1 = stochasticGradiantDescent(np.zeros(n), .1)
t2 = time.time_ns() - t2

plt.plot(np.arange(1000) * t1, cost1[:1000])
plt.plot(np.arange(1000) * t2, cost2[:1000], label='stochastic')
plt.legend()
```

[ ]: <matplotlib.legend.Legend at 0x117933370>



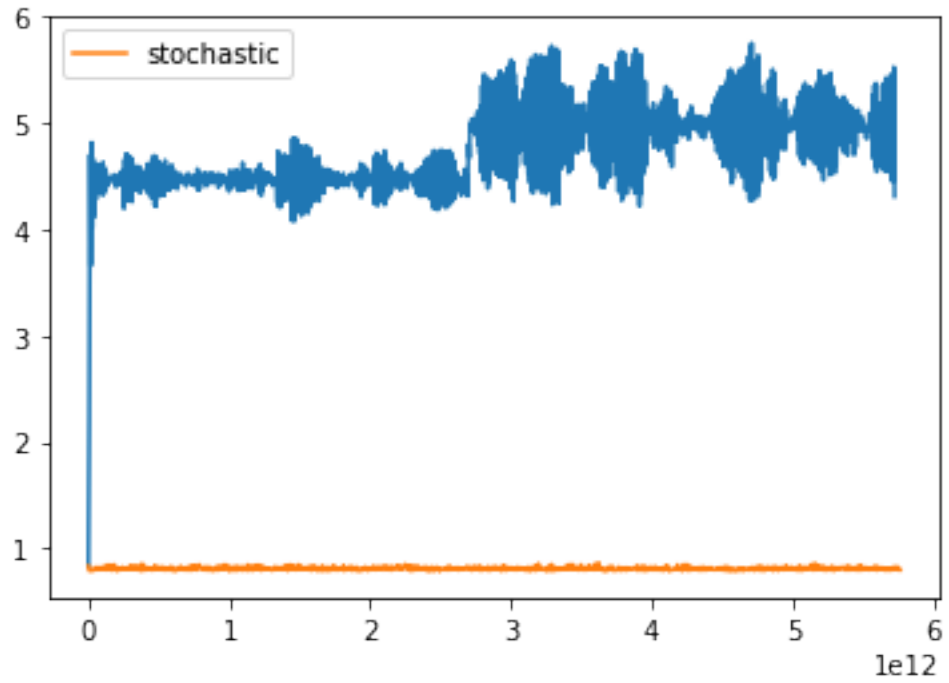
[ ]:
```
t1 = time.time_ns()
cost1 = gradiantDescent(np.zeros(n), 1)
t1 = time.time_ns() - t1

t2 = time.time_ns()
st_cost1 = stochasticGradiantDescent(np.zeros(n), 1)
t2 = time.time_ns() - t2

plt.plot(np.arange(1000) * t1, cost1[:1000])
plt.plot(np.arange(1000) * t2, cost2[:1000], label='stochastic')
plt.legend()
```

[ ]: <matplotlib.legend.Legend at 0x1178f91c0>



[ ]:
```
t1 = time.time_ns()
cost1 = gradiantDescent(np.zeros(n), 10)
t1 = time.time_ns() - t1

t2 = time.time_ns()
st_cost1 = stochasticGradiantDescent(np.zeros(n), 10)
t2 = time.time_ns() - t2

plt.plot(np.arange(1000) * t1, cost1[:1000])
plt.plot(np.arange(1000) * t2, cost2[:1000], label='stochastic')
plt.legend()
```

[ ]:
```
<matplotlib.legend.Legend at 0x117b7e220>
```



1.2 Q5)

[ ]:
```
import pandas
```

[ ]:
```
dataframe = pandas.read_csv('house_data.csv')
```

[ ]:
```
dataframe.head()
```

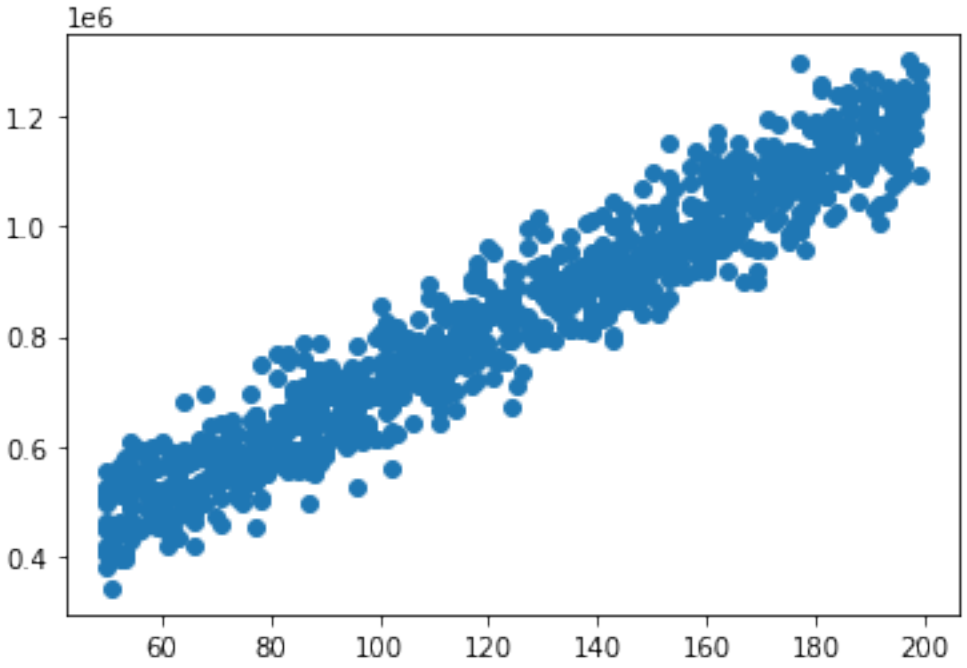
[ ]:

	size	bedrooms	bathrooms	age	Tehran	Esfehan	Yazd	Shiraz	Tabriz	\
0	152	1	3	3	0	0	0	0	1	
1	142	1	2	23	0	0	1	0	0	
2	64	1	2	26	0	0	0	0	1	
3	156	2	3	37	0	0	0	0	1	
4	121	4	2	24	0	0	1	0	0	

	price
0	981653.9936
1	870284.3499
2	515868.5989
3	979655.8731
4	859453.6265

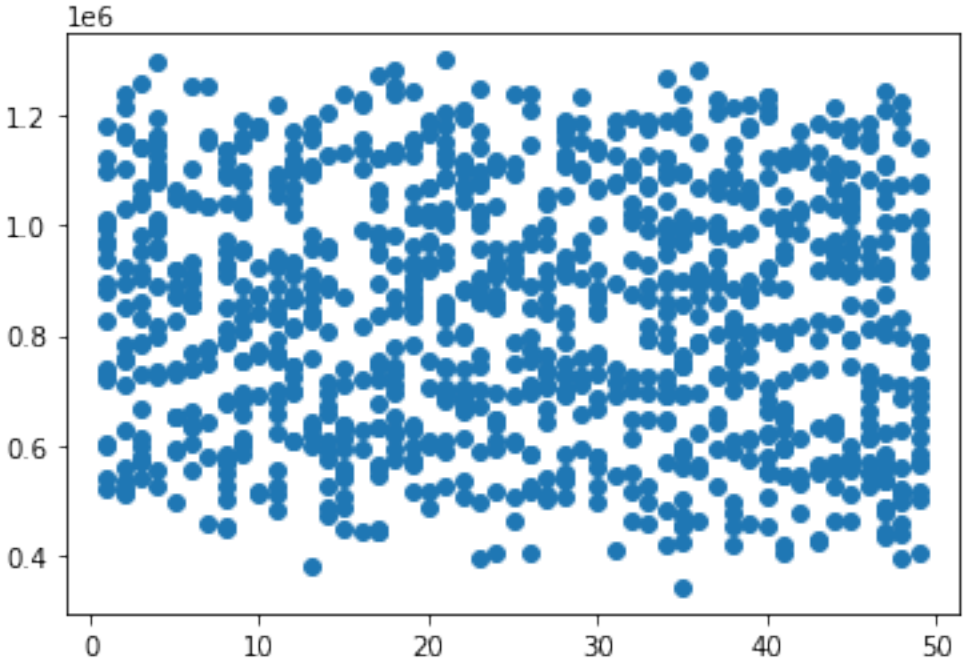
[ ]:
```
plt.scatter(dataframe['size'], dataframe.price)
```

[ ]: <matplotlib.collections.PathCollection at 0x121cf3d60>



[ ]:
```
plt.scatter(dataframe.age, dataframe.price)
```

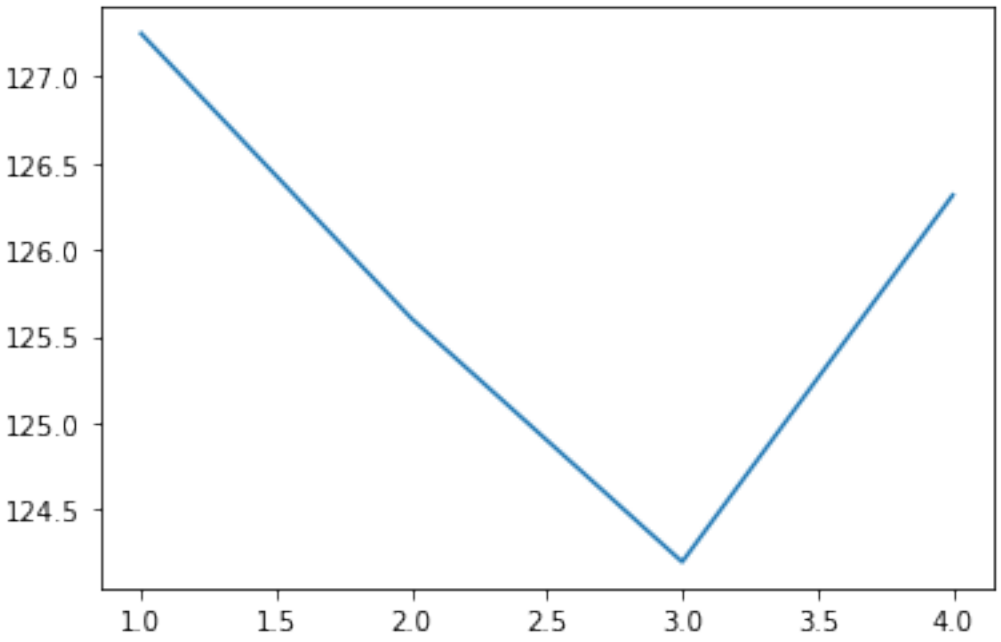
[ ]: <matplotlib.collections.PathCollection at 0x121d76700>



[ ]:
```
mean1 = dataframe.where(dataframe.bedrooms == 1)['size'].mean()
mean2 = dataframe.where(dataframe.bedrooms == 2)['size'].mean()
mean3 = dataframe.where(dataframe.bedrooms == 3)['size'].mean()
mean4 = dataframe.where(dataframe.bedrooms == 4)['size'].mean()
mean5 = dataframe.where(dataframe.bedrooms == 5)['size'].mean()
```

[ ]:
```
plt.plot([1,2,3,4,5], [mean1, mean2, mean3, mean4, mean5])
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x121ddc970>]
```



. . .

.

.

1.3 Q6)

[ ]:
```
train_size = (dataframe.__len__() // 3) * 2
test_size = dataframe.__len__() - train_size
```

[ ]:
```
trainData = dataframe.iloc[range(train_size)]
testData = dataframe.iloc[range(train_size, train_size + test_size)]
```

[ ]:
```
mean = trainData.mean()
std = trainData.std()
```

[ ]:
```
trainData -= mean
testData -= mean
```

[ ]:
```
trainData /= std
testData /= std
```

[ ]:
```
trainData.head()
```

[ ]:
```
       size  bedrooms  bathrooms       age    Tehran   Esfehan      Yazd  \
0  0.565367 -1.294139   1.312798 -1.594851 -0.513187 -0.494461 -0.487402
1  0.337427 -1.294139   0.074379 -0.205212 -0.513187 -0.494461  2.048613
2 -1.440507 -1.294139   0.074379  0.003234 -0.513187 -0.494461 -0.487402
3  0.656543 -0.393516   1.312798  0.767536 -0.513187 -0.494461 -0.487402
4 -0.141248  1.407731   0.074379 -0.135730 -0.513187 -0.494461  2.048613

     Shiraz    Tabriz     price
0 -0.522506  2.078822  0.607373
1 -0.522506 -0.480319  0.110404
2 -0.522506  2.078822 -1.471120
3 -0.522506  2.078822  0.598457
4 -0.522506 -0.480319  0.062074
```

1.4 Q7)

[ ]:
```
def lossFunction(prediction, true_value):
    return np.sum((prediction - true_value)**2) / prediction.shape[0]
```

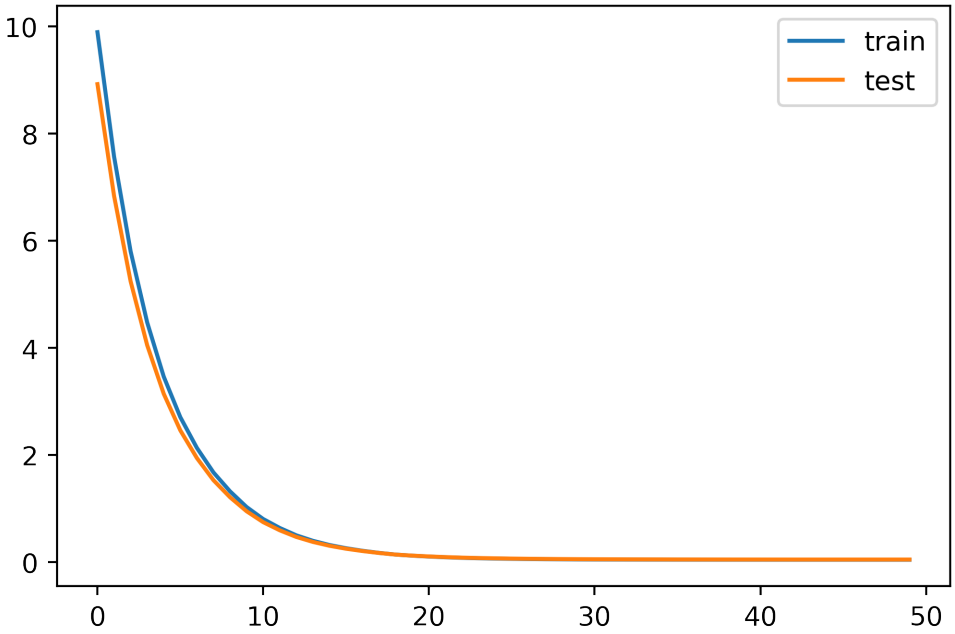
1.5 Q8)

[ ]:
```
def sgd(w, b, batch, eta):
    error = batch['X'] @ w + b - batch['Y']
    error /= batch['n']

    wGradiant = batch['X'].T @ error
    bGradiant = error.sum()

    return w - eta * wGradiant, b - eta * bGradiant
```

1.6 Q9)

[ ]:
```
numberOfEpochs = 50

batchSize = 60

w = np.random.normal(0,1,trainData.values.shape[1]-1)
b = 0


test = testData.values
testX = test[:,:-1]
testy = test[:,-1]

def eta(bNumber, epoch):
    return .01

trainLosses = []
testLosses = []
for epoch in range(numberOfEpochs):
    data = trainData.values
    np.random.shuffle(data)
    X = data[:,:-1]
    y = data[:,-1]

    numberOfBatches = int(np.ceil(data.shape[0] / batchSize))

    for bNumber in range(batchSize):
        l = bNumber * batchSize
        r = min(l + batchSize, data.shape[0])
        batch = {}
        batch['X'] = X[l:r]
        batch['Y'] = y[l:r]
        batch['n'] = r - l

        w, b = sgd(w, b, batch, eta(bNumber, epoch))

    trainLosses += [lossFunction(X @ w + b, y)]
    testLosses += [lossFunction(testX@w + b, testy)]



plt.figure(dpi=400)
plt.plot(range(numberOfEpochs), trainLosses, label='train')
plt.plot(range(numberOfEpochs), testLosses, label='test')

plt.legend()
```

```
plt.savefig('training.png')
```



[ ]: