Question 1

In [24]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split as split_data
from sklearn.metrics import mean_squared_error
```

In [25]:

```
def relu(z):
    a = np.maximum(0,z)
    return a
```

In [26]:

```
def initialize_params(layer_sizes):
    params = {}
    for i in range(1, len(layer_sizes)):
        params['W'+str(i)] = np.random.randn(layer_sizes[i], layer_sizes[i-1])*0.01
        params['B'+str(i)] = np.random.randn(layer_sizes[i],1)*0.01
    return params
```

In [27]:

```
def forward_propagation(X_train, params):
    layers = len(params)//2
    values = {}
    for i in range(1, layers+1):
        if i==1:
            values['Z'+str(i)] = np.dot(params['W'+str(i)], X_train) + params['B'+str(i)]
            values['A'+str(i)] = relu(values['Z'+str(i)])
        else:
            values['Z'+str(i)] = np.dot(params['W'+str(i)], values['A'+str(i-1)]) + params[
            values['A'+str(i)] = values['Z'+str(i)] if(i==layers) else relu(values['Z'+str(
    return values
```

In [28]:

```
def compute_cost(values, Y_train):
    layers = len(values)//2
    Y_pred = values['A'+str(layers)]
    cost = 1/(2*len(Y_train))*np.sum(np.square(Y_pred - Y_train))
    return cost
```

In [29]:

```
def backward_propagation(params, values, X_train, Y_train):
    layers = len(params)//2
    m = len(Y_train)
    grads = {}
    for i in range(layers,0,-1):
        if i==layers:
            dA = 1/m * (values['A'+str(i)] - Y_train)
            dZ = dA
        else:
            dA = np.dot(params['W'+str(i+1)].T, dZ)
            dZ = np.multiply(dA, np.where(values['A'+str(i)]>=0, 1, 0))
        grads['W' + str(i)] = 1/m * np.dot(dZ,values['A'+str(i-1)].T) if(i != 1) else (1/m)
        grads['B' + str(i)] = 1/m * np.sum(dZ, axis=1, keepdims=True)
    return grads
```

In [30]:

```
def update_params(params, grads, learning_rate):
    layers = len(params)//2
    params_updated = {}
    for i in range(1,layers+1):
        params_updated['W'+str(i)] = params['W'+str(i)] - learning_rate*grads['W'+str(i)]
        params_updated['B'+str(i)] = params['B'+str(i)] - learning_rate*grads['B'+str(i)]
    return params_updated
```

In [31]:

```
def model(X_train, Y_train, layer_sizes, num_iters, learning_rate):
    params = initialize_params(layer_sizes)
    for i in range(num_iters):
        values = forward_propagation(X_train.T, params)
        cost = compute_cost(values, Y_train.T)
        grads = backward_propagation(params, values,X_train.T, Y_train.T)
        params = update_params(params, grads, learning_rate)
#          print('Cost at iteration ' + str(i+1) + ' = ' + str(cost) + '\n')
    return params
```

In [32]:

```
def compute_accuracy(X_train, X_test, Y_train, Y_test, params, layer_sizes):
    values_train = forward_propagation(X_train.T, params)
    values_test = forward_propagation(X_test.T, params)
    train_acc = np.sqrt(mean_squared_error(Y_train, values_train['A'+str(len(layer_sizes)-1
    test_acc = np.sqrt(mean_squared_error(Y_test, values_test['A'+str(len(layer_sizes)-1)].
    return train_acc, test_acc
```

In [33]:

```
def predict(X, params):
    values = forward_propagation(X.T, params)
    predictions = values['A' + str(len(values)//2)].T
    return predictions
```

In [34]:

```
data = pd.read_csv('boston.csv', header=0)
```

In [35]:

```
#Question 1
#predict the values for Boston House prices dataset
num_iters = 1000
alpha = 0.03
layer_sizes=[13,6,5,1]
data = pd.read_csv('boston.csv')
n = data.shape[1]
m = data.shape[0]

x_train=np.array(df.iloc[:int(0.75*m),:n-1])
y_train=np.array(df.iloc[:int(0.75*m),-1])
x_test=np.array(df.iloc[int(0.75*m):,:n-1])
y_test=np.array(df.iloc[int(0.75*m):,-1])
params = model(x_train, y_train, layer_sizes, num_iters, alpha)
y_predict = predict(x_test,params)
print(y_predict)
```

```
 [24.69021328]
 [24.22888199]
 [20.35200618]
 [18.02034383]
 [23.99641773]
 [24.84234748]
 [24.5098568 ]
 [24.19449608]
 [23.30672943]
 [23.56340476]
 [24.42750703]
 [24.52775754]
 [24.26168652]
 [24.50806254]
 [24.302818  ]
 [24.75544498]
 [25.36560798]
 [25.01879362]
 [24.94537205]]
```

Question 2

In [36]:

```
#predict the values for Boston House prices dataset
num_iters = 1000
alpha = 0.03
layer_sizes=[7,5,5,1]
data =  pd.read_csv('seeds.csv')
n = data.shape[1]
m = data.shape[0]

x_train=np.array(df.iloc[:int(0.8*m),:n-1])
y_train=np.array(df.iloc[:int(0.8*m),-1])
x_test=np.array(df.iloc[int(0.8*m):,:n-1])
y_test=np.array(df.iloc[int(0.8*m):,-1])
params = model(x_train, y_train, layer_sizes, num_iters, alpha)
y_predict = predict(x_test,params)
print(y_predict)
```

```
 [16.62374057]
 [19.15646331]
 [19.38024364]
 [18.71016088]
 [20.57317848]
 [19.7222397 ]
 [16.93523927]
 [16.75901773]
 [15.52350192]
 [17.29903092]
 [17.9918474 ]
 [18.16021906]
 [17.71620992]
 [18.97684605]
 [19.64447219]
 [19.10715487]
 [21.62809731]
 [21.156228  ]
 [19.1526005 ]]
```

In [ ]: