HOMEWORK 1_801333188

hw1-1a

February 20, 2023

[17]:
```
import numpy as np # import numpy library
import pandas as pd # import pandas library
import matplotlib.pyplot as plt # import matplotlib library
from sklearn import preprocessing  # import scikit-learn library (source: https:
 ↪//scikit-learn.org/stable/index.html)
```

[18]:
```
df = pd.read_csv("/content/sample_data/Housing.csv")

# display DataFrame
df
```

[18]:
```
        price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0    13300000  7420         4          2        3      yes        no       no
1    12250000  8960         4          4        4      yes        no       no
2    12250000  9960         3          2        2      yes        no      yes
3    12215000  7500         4          2        2      yes        no      yes
4    11410000  7420         4          1        2      yes       yes      yes
..        …   …       …        …      …      …        …
540   1820000  3000         2          1        1      yes        no      yes
541   1767150  2400         3          1        1       no        no       no
542   1750000  3620         2          1        1      yes        no       no
543   1750000  2910         3          1        1       no        no       no
544   1750000  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
0                no             yes        2      yes        furnished
1                no             yes        3       no        furnished
2                no              no        2      yes   semi-furnished
3                no             yes        3      yes        furnished
4                no             yes        2       no        furnished
..              …             …      …      …              …
540              no              no        2       no      unfurnished
541              no              no        0       no   semi-furnished
542              no              no        0       no      unfurnished
543              no              no        0       no        furnished
544              no              no        0       no      unfurnished
```

```
[545 rows x 13 columns]
```

[19]:
```
df = df.replace(to_replace=['yes', 'no'], value=[1, 0])
df
```

[19]:
```
        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0    13300000  7420         4          2        3         1          0
1    12250000  8960         4          4        4         1          0
2    12250000  9960         3          2        2         1          0
3    12215000  7500         4          2        2         1          0
4    11410000  7420         4          1        2         1          1
..        …   …       …        …      …       …        …
540   1820000  3000         2          1        1         1          0
541   1767150  2400         3          1        1         0          0
542   1750000  3620         2          1        1         1          0
543   1750000  2910         3          1        1         0          0
544   1750000  3850         3          1        2         1          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
0           0                0                1        2         1
1           0                0                1        3         0
2           1                0                0        2         1
3           1                0                1        3         1
4           1                0                1        2         0
..        …               …               …      …       …
540         1                0                0        2         0
541         0                0                0        0         0
542         0                0                0        0         0
543         0                0                0        0         0
544         0                0                0        0         0

    furnishingstatus
0          furnished
1          furnished
2     semi-furnished
3          furnished
4          furnished
..               …
540      unfurnished
541   semi-furnished
542      unfurnished
543        furnished
544      unfurnished

[545 rows x 13 columns]
```

[20]:

```
# create a training set by randomly selecting 80% of the rows from the DataFrame
train = df.sample(frac=0.8, random_state=1)

# create a test set by dropping the rows in the training set from the DataFrame
test = df.drop(train.index)
```

[21]:

```
test
```

[21]:

```
        price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
2    12250000   9960         3          2        2         1          0
3    12215000   7500         4          2        2         1          0
7    10150000  16200         5          3        2         1          0
15    9100000   6000         4          1        2         1          0
22    8645000   8050         3          1        1         1          1
..        …      …        …         …       …         …
508   2590000   4400         2          1        1         1          0
513   2485000   4400         3          1        2         1          0
520   2450000   7700         2          1        1         1          0
537   1890000   1700         3          1        2         1          0
539   1855000   2990         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
2           1                0                0        2         1
3           1                0                1        3         1
7           0                0                0        0         0
15          1                0                0        2         0
22          1                0                1        1         0
..        …               …               …       …        …
508         0                0                0        0         0
513         0                0                0        0         0
520         0                0                0        0         0
537         0                0                0        0         0
539         0                0                0        1         0

    furnishingstatus
2     semi-furnished
3          furnished
7        unfurnished
15    semi-furnished
22         furnished
..               …
508      unfurnished
513      unfurnished
520      unfurnished
537      unfurnished
539      unfurnished
```

```
[109 rows x 13 columns]
```

[22]:
```
# select specific columns for the training set
train = train[['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']]

# select specific columns for the test set
test = test[['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']]
```

[23]:
```
import sklearn.preprocessing # import scikit-learn library for data␣
 ↪preprocessing

# create an instance of the MinMaxScaler class for scaling features to a range␣
 ↪of [0, 1]
scaler = sklearn.preprocessing.MinMaxScaler()
```

[24]:
```
train
```

[24]:
```
       price   area  bedrooms  bathrooms  stories  parking
62   7070000   6240         4          2        2        1
247  4550000   8400         4          1        4        3
142  5600000  10500         4          2        2        1
107  6125000   6420         3          1        3        0
483  2940000   6615         3          1        2        0
..       …      …         …          …        …        …
359  3710000   3600         3          1        1        1
36   8043000   7482         3          2        3        1
30   8400000   7475         3          2        4        2
20   8750000   4320         3          1        2        2
527  2275000   1836         2          1        1        0

[436 rows x 6 columns]
```

[25]:
```
# create a NumPy array of the 'area' column from the training set
X1_t = np.array(train['area'])

# display the NumPy array
X1_t
```

[25]:
```
array([ 6240,  8400, 10500,  6420,  6615,  3600,  3240,  6600,  2700,
        5000,  2650,  4775,  4800,  3700,  7700,  7420,  4280,  6000,
        6600,  3649,  3420,  5500,  3630,  3180,  3600,  8400,  3000,
        8880,  5750,  2145,  6360,  6525,  1950,  5850,  8372,  2870,
        4990,  2684,  5200,  6321,  4960,  3480,  3210,  4950,  6840,
        4350,  5850,  4410,  2500,  3850,  3180,  3162,  3500,  4340,
        6440,  5010,  3000,  4920,  3760,  3816,  6000,  7000,  3640,
        4080,  4160,  2910,  6060,  3000,  2787,  4815,  4785,  6600,
        5300,  3600,  6000,  2176,  3000,  7420,  7020,  3480,  5960,
        3510,  6420,  6450,  6210,  4500,  3000,  3180,  5700,  3520,
        4040,  5800,  2800,  6480,  4960,  4260,  7500,  5880, 10500,
        4500,  3850,  8500,  3120,  3990,  4095,  4800, 13200,  7770,
        6100,  4075,  6550,  4100,  4370,  3180,  7350,  3510,  3640,
        5500,  8250,  6600,  8250,  2475,  3850,  4500,  3720,  4360,
       10240,  5500,  3970,  3450,  3850,  5500,  3520,  2145,  6600,
        3640,  3986,  2953,  8250,  4130,  8580,  6000,  3500,  5885,
        7680,  2430,  3150,  6450,  8100,  5500,  1650,  3040,  4079,
        2747,  4600,  2325,  7231,  3520,  2145,  3450,  3620,  4000,
        6000,  6000,  4500,  3540,  7200,  3120,  4000,  2015,  4040,
        8000,  2787,  3512,  3420,  6060,  4500,  6360,  5450,  8250,
        3960,  7410, 10360,  3630,  6020,  4100,  6254,  4500,  4560,
        6710,  3500,  8880,  3600,  7152,  6000,  4040,  4000,  4040,
        5360,  6600,  3800,  3960,  4900,  3480,  3584,  2275,  4000,
        6500, 10500,  8960,  3290,  8875,  8580,  3450,  6600,  2800,
        5640,  3745, 10269,  6100, 12090,  5880,  6750,  6000,  5320,
        4000,  4040, 15600,  3090,  3970,  5450,  4770,  4095,  6000,
        6540,  6550,  4320,  3100,  4050,  3650,  3850,  5600,  2817,
        4510,  3000,  4995, 11410,  3000,  4840,  3600,  4000,  3500,
        7800,  5300,  4840,  3000,  3480,  2970,  5828,  3800,  4040,
       10700,  7320,  5000,  6325,  2880,  4300,  3150,  4000,  9500,
        4500,  3420,  3180,  2145,  5400,  3630,  6750,  4820,  5136,
        4120,  6825,  4600,  6650,  5800,  5720,  5000,  4352,  3300,
        2160,  5900,  3000,  4500,  3350,  5400,  4600,  9800,  3630,
        2610,  9667,  3635,  4000,  3180,  3630,  6600,  2610,  4960,
        5150,  6000,  3640,  2910,  3650,  3450,  4032,  7980,  1905,
        6000,  3360,  9620,  1950, 12900,  3240,  4320,  6540,  6000,
        7440,  3760,  8100,  4880,  6000,  2000,  5200,  4050,  9166,
        7950,  5500,  2700,  6000,  6900,  3500,  5076,  5985,  4300,
        8050,  5320,  5960,  7000,  7260,  6360,  3000,  3460, 12944,
        3880,  2400,  4080,  6000,  4500,  6050,  7000,  3930,  4600,
        7155,  4100,  2400,  3460,  4632,  4200,  4640,  8800,  3000,
        6300,  7000,  7000,  6900,  3420,  3264,  2640,  3150,  4320,
        6862, 11440,  4992,  3069,  3185,  3750,  5300,  7200,  6400,
        6800,  3400,  6420,  3792,  5500,  4600,  6800,  6000,  8520,
        6480,  8150,  5948,  3185,  5830,  3410,  3000,  8400,  6350,
        8100,  4800,  2856,  3185,  3780,  3640,  6000,  6000,  4800,
        5800,  6360,  4120,  5400,  2850,  5400,  2145,  4500,  3240,
       13200,  3900,  9000,  4646,  3840,  9000,  3520,  3640,  3600,
        7482,  7475,  4320,  1836])
```

[26]:

```
X1_t = np.array(train['area']) # 'area' column
X2_t = np.array(train['bedrooms']) # 'bedrooms' column
X3_t = np.array(train['bathrooms']) # 'bathrooms' column
X4_t = np.array(train['stories']) # 'stories' column
X5_t = np.array(train['parking']) # 'parking' column
```

```
# create a NumPy array of ones to represent the bias term
X0_t = np.ones(len(train))
```

```
[27]: # stack the selected feature arrays vertically using np.vstack
X = np.vstack([X0_t, X1_t, X2_t, X3_t, X4_t, X5_t])

# transpose the stacked array to make it a 6 x 436 matrix
X = X.T

# convert the stacked array to a NumPy array
X = np.array(X)

# display the NumPy array
X
```

```
[27]: array([[1.000e+00, 6.240e+03, 4.000e+00, 2.000e+00, 2.000e+00, 1.000e+00],
       [1.000e+00, 8.400e+03, 4.000e+00, 1.000e+00, 4.000e+00, 3.000e+00],
       [1.000e+00, 1.050e+04, 4.000e+00, 2.000e+00, 2.000e+00, 1.000e+00],
       …,
       [1.000e+00, 7.475e+03, 3.000e+00, 2.000e+00, 4.000e+00, 2.000e+00],
       [1.000e+00, 4.320e+03, 3.000e+00, 1.000e+00, 2.000e+00, 2.000e+00],
       [1.000e+00, 1.836e+03, 2.000e+00, 1.000e+00, 1.000e+00, 0.000e+00]])
```

```
[28]: # scale the feature matrix using the fit_transform() method of the scaler object
X_scaled = scaler.fit_transform(X)

# assign the scaled feature matrix to the original variable name 'X'
X = X_scaled
```

```
[29]: # create a 1D NumPy array of zeros with length 6
theta = np.zeros(6)

# reshape the 1D array to a column vector using np.reshape
theta = np.reshape(theta, (6,1))

# display the column vector
theta
```

```
[29]: array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

```
[30]: # create a 1D NumPy array 'Y_t' from the 'price' column of the training set␣
  ↪'train'
```

```
Y_t = np.array(train.price)

# create a copy of 'Y_t' to prevent changing the original data
Y = Y_t.copy()

# reshape 'Y' to a column vector using np.reshape
Y = np.reshape(Y, (436,1))

# create an instance of the MinMaxScaler class for scaling the target variable␣
 ↪to a range of [0, 1]
scaler = sklearn.preprocessing.MinMaxScaler()

# scale the target variable using the fit_transform() method of the scaler␣
 ↪object
Y_scaled = scaler.fit_transform(Y)

# assign the scaled target variable to the original variable name 'Y'
Y = Y_scaled
```

[31]:

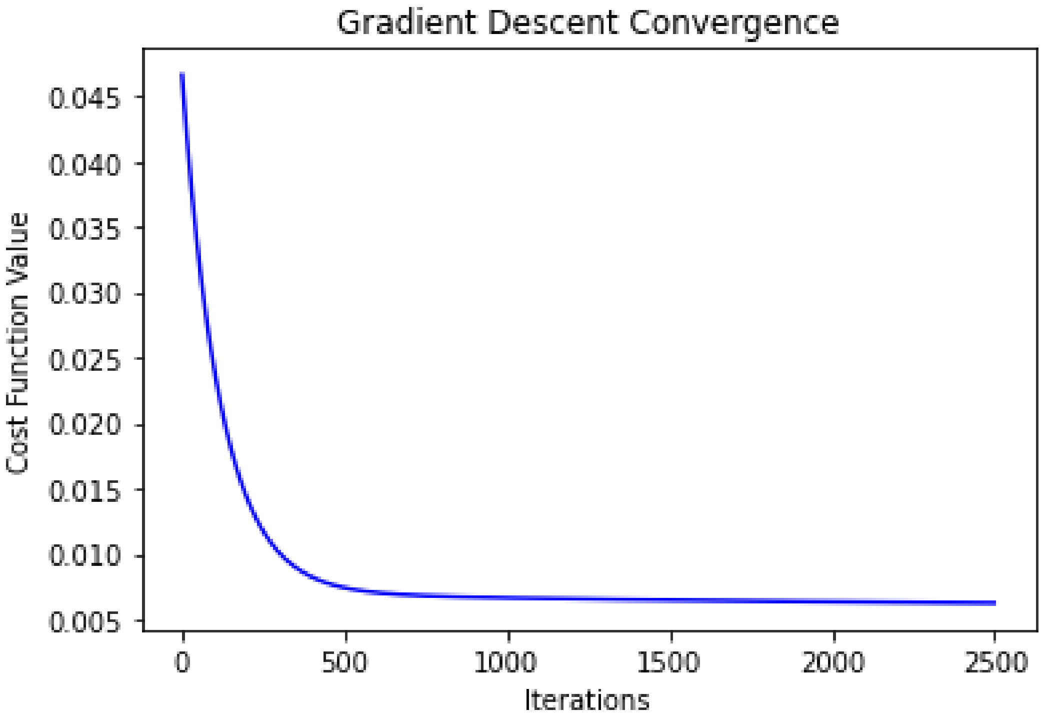
```
# create a NumPy array 'X_T' containing the transpose of the feature matrix 'X'
X_T = np.array(X.T)

# retrieve the number of rows 'm' and the number of columns 'n' from the␣
 ↪feature matrix 'X'
m, n = X.shape

# display the values of 'm' and 'n'
print("Number of training examples (m): ", m)
print("Number of features (n): ", n)

# set the number of iterations for gradient descent
iterations = 2500

# create a counter variable 'count' and a NumPy array 'j' to store the cost␣
 ↪function values for each iteration
count = 0
j = np.zeros(shape=(iterations, 1), dtype=float)

# display the shape of the 'j' array
print("Shape of 'j' array: ", j.shape)
```

```
Number of training examples (m):  436
Number of features (n):  6
Shape of 'j' array:  (2500, 1)
```

[32]:

```
# set the initial iteration count to zero
count = 0
```

```
# create a NumPy array 'j' to store the cost function values for each iteration
j = np.zeros(shape=(iterations, 1), dtype=float)

# perform gradient descent for the specified number of iterations
while count < iterations:

    # calculate the predicted values 'h' using the current parameters 'theta'
    h = X.dot(theta)

    # calculate the cost function value 'j' using the current parameters 'theta'
    j[count] = (1/(2*m)) * np.sum((h-Y)**2)

    # calculate the gradient of the cost function with respect to 'theta'
    grad = (1/m) * X_T.dot(h-Y)

    # update the parameters 'theta' using the learning rate 'alpha' and the␣
 ↪gradient 'grad'
    alpha = 0.01
    theta = theta - alpha * grad

    # increment the iteration count
    count += 1

# plot the cost function values over the iterations
plt.plot(j,'b-')
plt.xlabel('Iterations')
plt.ylabel('Cost Function Value')
plt.title('Gradient Descent Convergence')
plt.show()
```



[33]:

```
# extract the test set features into NumPy arrays
N_X1_t = np.array(test.area)
N_X2_t = np.array(test.bedrooms)
N_X3_t = np.array(test.bathrooms)
N_X4_t = np.array(test.stories)
N_X5_t = np.array(test.parking)
N_X0_t = np.ones(109)
```

[34]:

```
# stack the test set features into a design matrix
N_X = np.vstack([N_X0_t,N_X1_t,N_X2_t,N_X3_t,N_X4_t,N_X5_t])
N_X_T = N_X.T
N_X = np.array(N_X_T)
N_x = scaler.fit_transform(N_X)
N_X = N_x
N_X.shape
```

[34]:

```
(109, 6)
```

[35]:

```
N_theta = np.array([0.,0.,0.,0.,0,0.])
N_theta = N_theta.reshape(6,1)
N_theta
```

```
[35]: array([[0.],
              [0.],
              [0.],
              [0.],
              [0.],
              [0.]])
```

```
[36]: N_Y_t = np.array(test.price)
      N_Y = N_Y_t
      N_Y = N_Y_t.reshape(109,1)
      N_y = scaler.fit_transform(N_Y)
      N_Y=N_y
      N_Y.shape
```

```
[36]: (109, 1)
```

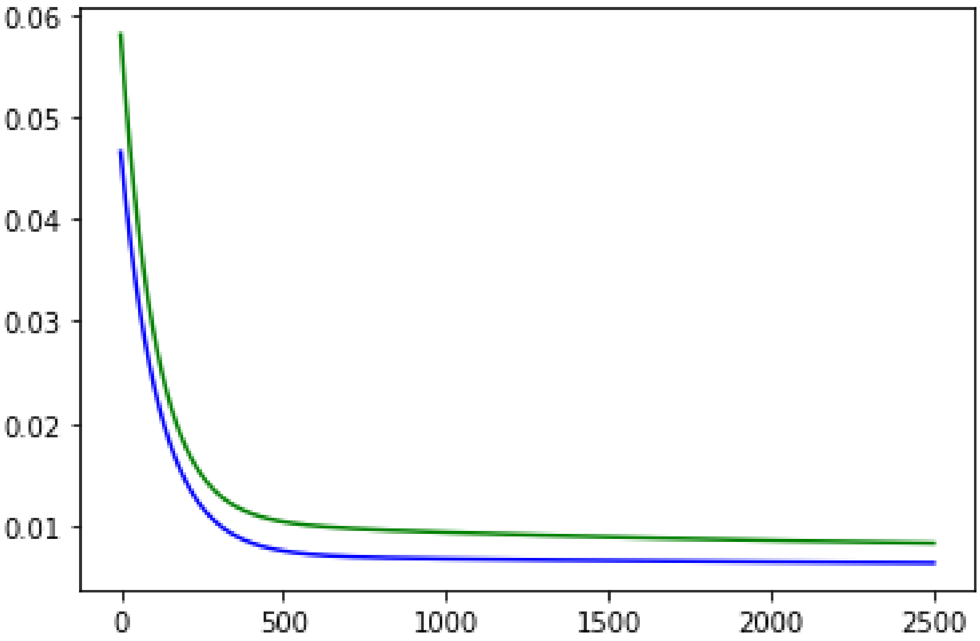
```
[37]: N_X_T = np.array(N_X.T)
      m,n = N_X.shape
      m,n
```

```
[37]: (109, 6)
```

```
[38]: iterations = 2500
      count=0
      N_j = np.zeros(shape=(iterations, 1), dtype=float)

      while(count < iterations):

          N_h_t = N_X.dot(N_theta)
          N_h = np.array(N_h_t)


          N_j[count]= (1/(2*m))*np.sum((N_h - N_Y)**2)

          grad_t = N_X_T.dot(N_h-N_Y)
          grad = grad_t*(1/m)

          N_theta = N_theta - 0.01*(grad)

          count += 1
```

```
[39]: plt.plot(N_j,'g-')
      plt.plot(j,'b-')
```

```
[39]: [<matplotlib.lines.Line2D at 0x7f84302065e0>]
```


0.06
0.05
0.04
0.03
0.02
0.01
0
500
1000
1500
2000
2500

hw1-1b

February 20, 2023

[1]:
```
import numpy as np # import numpy library
import pandas as pd # import pandas library
import matplotlib.pyplot as plt # import matplotlib library
from sklearn import preprocessing  # import scikit-learn library (source: https:
 ↪//scikit-learn.org/stable/index.html)
```

[2]:
```
df = pd.read_csv("/content/sample_data/Housing.csv")

# display DataFrame
df
```

[2]:
```
        price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0    13300000  7420         4          2        3      yes        no       no
1    12250000  8960         4          4        4      yes        no       no
2    12250000  9960         3          2        2      yes        no      yes
3    12215000  7500         4          2        2      yes        no      yes
4    11410000  7420         4          1        2      yes       yes      yes
..        …   …       …        …      …      …       …      …
540   1820000  3000         2          1        1      yes        no      yes
541   1767150  2400         3          1        1       no        no       no
542   1750000  3620         2          1        1      yes        no       no
543   1750000  2910         3          1        1       no        no       no
544   1750000  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
0                no             yes        2      yes        furnished
1                no             yes        3       no        furnished
2                no              no        2      yes   semi-furnished
3                no             yes        3      yes        furnished
4                no             yes        2       no        furnished
..              …             …      …      …              …
540              no              no        2       no      unfurnished
541              no              no        0       no   semi-furnished
542              no              no        0       no      unfurnished
543              no              no        0       no        furnished
544              no              no        0       no      unfurnished
```

```
[545 rows x 13 columns]
```

```
[3]: df = df.replace(to_replace=['yes', 'no'], value=[1, 0])
     df
```

```
[3]:        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
     0    13300000  7420         4          2        3         1          0
     1    12250000  8960         4          4        4         1          0
     2    12250000  9960         3          2        2         1          0
     3    12215000  7500         4          2        2         1          0
     4    11410000  7420         4          1        2         1          1
     ..        …   …         …        …      …        …        …
     540   1820000  3000         2          1        1         1          0
     541   1767150  2400         3          1        1         0          0
     542   1750000  3620         2          1        1         1          0
     543   1750000  2910         3          1        1         0          0
     544   1750000  3850         3          1        2         1          0

          basement  hotwaterheating  airconditioning  parking  prefarea  \
     0           0                0                1        2         1
     1           0                0                1        3         0
     2           1                0                0        2         1
     3           1                0                1        3         1
     4           1                0                1        2         0
     ..        …              …              …      …       …
     540         1                0                0        2         0
     541         0                0                0        0         0
     542         0                0                0        0         0
     543         0                0                0        0         0
     544         0                0                0        0         0

         furnishingstatus
     0          furnished
     1          furnished
     2     semi-furnished
     3          furnished
     4          furnished
     ..               …
     540      unfurnished
     541   semi-furnished
     542      unfurnished
     543        furnished
     544      unfurnished

     [545 rows x 13 columns]
```

[4]:
```
# create a training set by randomly selecting 80% of the rows from the DataFrame
train = df.sample(frac=0.8, random_state=1)

# create a test set by dropping the rows in the training set from the DataFrame
test = df.drop(train.index)
```

[5]:
```
train
```

[5]:
```
       price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
62   7070000   6240         4          2        2         1          0
247  4550000   8400         4          1        4         1          0
142  5600000  10500         4          2        2         1          0
107  6125000   6420         3          1        3         1          0
483  2940000   6615         3          1        2         1          0
..       …      …       …         …        …        …         …
359  3710000   3600         3          1        1         1          0
36   8043000   7482         3          2        3         1          0
30   8400000   7475         3          2        4         1          0
20   8750000   4320         3          1        2         1          0
527  2275000   1836         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
62          0                0                1        1         0
247         0                0                0        3         0
142         0                0                0        1         0
107         1                0                0        0         1
483         0                0                0        0         0
..        …              …               …      …        …
359         0                0                0        1         0
36          0                1                0        1         1
30          0                0                1        2         0
20          1                1                0        2         0
527         1                0                0        0         0

    furnishingstatus
62         furnished
247      unfurnished
142   semi-furnished
107      unfurnished
483   semi-furnished
..               …
359      unfurnished
36         furnished
30       unfurnished
20    semi-furnished
527   semi-furnished
```

```
[436 rows x 13 columns]
```

[6]:

```
test
```

[6]:

```
        price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
2    12250000   9960         3          2        2         1          0
3    12215000   7500         4          2        2         1          0
7    10150000  16200         5          3        2         1          0
15    9100000   6000         4          1        2         1          0
22    8645000   8050         3          1        1         1          1
..        …    …       …        …       …       …        …
508   2590000   4400         2          1        1         1          0
513   2485000   4400         3          1        2         1          0
520   2450000   7700         2          1        1         1          0
537   1890000   1700         3          1        2         1          0
539   1855000   2990         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
2           1                0                0        2         1
3           1                0                1        3         1
7           0                0                0        0         0
15          1                0                0        2         0
22          1                0                1        1         0
..        …              …              …      …       …
508         0                0                0        0         0
513         0                0                0        0         0
520         0                0                0        0         0
537         0                0                0        0         0
539         0                0                0        1         0

    furnishingstatus
2     semi-furnished
3          furnished
7        unfurnished
15    semi-furnished
22         furnished
..               …
508      unfurnished
513      unfurnished
520      unfurnished
537      unfurnished
539      unfurnished

[109 rows x 13 columns]
```

[7]:

```
# select specific columns for the training set
```

```
train = train[['price','area','bedrooms','bathrooms','stories','parking',␣
 ↪'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',␣
 ↪'prefarea']]
# select specific columns for the test set
test = test[['price','area','bedrooms','bathrooms','stories','parking',␣
 ↪'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',␣
 ↪'prefarea']]
```

[8]:
```
import sklearn.preprocessing # import scikit-learn library for data␣
 ↪preprocessing

# create an instance of the MinMaxScaler class for scaling features to a range␣
 ↪of [0, 1]
scaler = sklearn.preprocessing.MinMaxScaler()
```

[9]:
```
train
```

[9]:
```
       price   area  bedrooms  bathrooms  stories  parking  mainroad  \
62   7070000   6240         4          2        2        1         1
247  4550000   8400         4          1        4        3         1
142  5600000  10500         4          2        2        1         1
107  6125000   6420         3          1        3        0         1
483  2940000   6615         3          1        2        0         1
..       …      …         …          …        …        …         …
359  3710000   3600         3          1        1        1         1
36   8043000   7482         3          2        3        1         1
30   8400000   7475         3          2        4        2         1
20   8750000   4320         3          1        2        2         1
527  2275000   1836         2          1        1        0         0

     guestroom  basement  hotwaterheating  airconditioning  prefarea
62           0         0                0                1         0
247          0         0                0                0         0
142          0         0                0                0         0
107          0         1                0                0         1
483          0         0                0                0         0
..         …         …              …                …         …
359          0         0                0                0         0
36           0         0                1                0         1
30           0         0                0                1         0
20           0         1                1                0         0
527          0         1                0                0         0

[436 rows x 12 columns]
```

[10]:

```
# create a NumPy array of the 'area' column from the training set
X1_t = np.array(train['area'])

# display the NumPy array
X1_t
```

[10]:

```
array([ 6240,  8400, 10500,  6420,  6615,  3600,  3240,  6600,  2700,
        5000,  2650,  4775,  4800,  3700,  7700,  7420,  4280,  6000,
        6600,  3649,  3420,  5500,  3630,  3180,  3600,  8400,  3000,
        8880,  5750,  2145,  6360,  6525,  1950,  5850,  8372,  2870,
        4990,  2684,  5200,  6321,  4960,  3480,  3210,  4950,  6840,
        4350,  5850,  4410,  2500,  3850,  3180,  3162,  3500,  4340,
        6440,  5010,  3000,  4920,  3760,  3816,  6000,  7000,  3640,
        4080,  4160,  2910,  6060,  3000,  2787,  4815,  4785,  6600,
        5300,  3600,  6000,  2176,  3000,  7420,  7020,  3480,  5960,
        3510,  6420,  6450,  6210,  4500,  3000,  3180,  5700,  3520,
        4040,  5800,  2800,  6480,  4960,  4260,  7500,  5880, 10500,
        4500,  3850,  8500,  3120,  3990,  4095,  4800, 13200,  7770,
        6100,  4075,  6550,  4100,  4370,  3180,  7350,  3510,  3640,
        5500,  8250,  6600,  8250,  2475,  3850,  4500,  3720,  4360,
       10240,  5500,  3970,  3450,  3850,  5500,  3520,  2145,  6600,
        3640,  3986,  2953,  8250,  4130,  8580,  6000,  3500,  5885,
        7680,  2430,  3150,  6450,  8100,  5500,  1650,  3040,  4079,
        2747,  4600,  2325,  7231,  3520,  2145,  3450,  3620,  4000,
        6000,  6000,  4500,  3540,  7200,  3120,  4000,  2015,  4040,
        8000,  2787,  3512,  3420,  6060,  4500,  6360,  5450,  8250,
        3960,  7410, 10360,  3630,  6020,  4100,  6254,  4500,  4560,
        6710,  3500,  8880,  3600,  7152,  6000,  4040,  4000,  4040,
        5360,  6600,  3800,  3960,  4900,  3480,  3584,  2275,  4000,
        6500, 10500,  8960,  3290,  8875,  8580,  3450,  6600,  2800,
        5640,  3745, 10269,  6100, 12090,  5880,  6750,  6000,  5320,
        4000,  4040, 15600,  3090,  3970,  5450,  4770,  4095,  6000,
        6540,  6550,  4320,  3100,  4050,  3650,  3850,  5600,  2817,
        4510,  3000,  4995, 11410,  3000,  4840,  3600,  4000,  3500,
        7800,  5300,  4840,  3000,  3480,  2970,  5828,  3800,  4040,
       10700,  7320,  5000,  6325,  2880,  4300,  3150,  4000,  9500,
        4500,  3420,  3180,  2145,  5400,  3630,  6750,  4820,  5136,
        4120,  6825,  4600,  6650,  5800,  5720,  5000,  4352,  3300,
        2160,  5900,  3000,  4500,  3350,  5400,  4600,  9800,  3630,
        2610,  9667,  3635,  4000,  3180,  3630,  6600,  2610,  4960,
        5150,  6000,  3640,  2910,  3650,  3450,  4032,  7980,  1905,
        6000,  3360,  9620,  1950, 12900,  3240,  4320,  6540,  6000,
        7440,  3760,  8100,  4880,  6000,  2000,  5200,  4050,  9166,
        7950,  5500,  2700,  6000,  6900,  3500,  5076,  5985,  4300,
        8050,  5320,  5960,  7000,  7260,  6360,  3000,  3460, 12944,
        3880,  2400,  4080,  6000,  4500,  6050,  7000,  3930,  4600,
        7155,  4100,  2400,  3460,  4632,  4200,  4640,  8800,  3000,
        6300,  7000,  7000,  6900,  3420,  3264,  2640,  3150,  4320,
        6862, 11440,  4992,  3069,  3185,  3750,  5300,  7200,  6400,
        6800,  3400,  6420,  3792,  5500,  4600,  6800,  6000,  8520,
        6480,  8150,  5948,  3185,  5830,  3410,  3000,  8400,  6350,
        8100,  4800,  2856,  3185,  3780,  3640,  6000,  6000,  4800,
        5800,  6360,  4120,  5400,  2850,  5400,  2145,  4500,  3240,
       13200,  3900,  9000,  4646,  3840,  9000,  3520,  3640,  3600,
        7482,  7475,  4320,  1836])
```

[11]:
```
X2_t = np.array(train.bedrooms)
X3_t = np.array(train.bathrooms)
X4_t = np.array(train.stories)
X5_t = np.array(train.parking)
X6_t = np.array(train.mainroad)
X7_t = np.array(train.guestroom)
X8_t = np.array(train.hotwaterheating)
X9_t = np.array(train.airconditioning)
X10_t = np.array(train.prefarea)
X11_t = np.array(train.basement)
X0_t= np.ones(436)
```

[12]:
```
# stack the selected feature arrays vertically using np.vstack
X = np.vstack([X0_t, X1_t, X2_t, X3_t, X4_t, X5_t,␣
 ↪X6_t,X7_t,X8_t,X9_t,X10_t,X11_t])

# transpose the stacked array to make it a 6 x 436 matrix
X = X.T

# convert the stacked array to a NumPy array
X = np.array(X)

# display the NumPy array
X
```

[12]:
```
array([[1.000e+00, 6.240e+03, 4.000e+00, …, 1.000e+00, 0.000e+00,
        0.000e+00],
       [1.000e+00, 8.400e+03, 4.000e+00, …, 0.000e+00, 0.000e+00,
        0.000e+00],
       [1.000e+00, 1.050e+04, 4.000e+00, …, 0.000e+00, 0.000e+00,
        0.000e+00],
       …,
       [1.000e+00, 7.475e+03, 3.000e+00, …, 1.000e+00, 0.000e+00,
        0.000e+00],
       [1.000e+00, 4.320e+03, 3.000e+00, …, 0.000e+00, 0.000e+00,
        1.000e+00],
       [1.000e+00, 1.836e+03, 2.000e+00, …, 0.000e+00, 0.000e+00,
        1.000e+00]])
```

[13]:

```
# scale the feature matrix using the fit_transform() method of the scaler object
X_scaled = scaler.fit_transform(X)

# assign the scaled feature matrix to the original variable name 'X'
X = X_scaled
```

[14]:

```
# create a 1D NumPy array of zeros with length 12
theta = np.zeros(12)

# reshape the 1D array to a column vector using np.reshape
theta = np.reshape(theta, (12,1))

# display the column vector
theta
```

[14]:

```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[15]:

```
# create a 1D NumPy array 'Y_t' from the 'price' column of the training set␣
  ↪'train'
Y_t = np.array(train.price)

# create a copy of 'Y_t' to prevent changing the original data
Y = Y_t.copy()

# reshape 'Y' to a column vector using np.reshape
Y = np.reshape(Y, (436,1))

# create an instance of the MinMaxScaler class for scaling the target variable␣
  ↪to a range of [0, 1]
scaler = sklearn.preprocessing.MinMaxScaler()

# scale the target variable using the fit_transform() method of the scaler␣
  ↪object
Y_scaled = scaler.fit_transform(Y)

# assign the scaled target variable to the original variable name 'Y'
```

```
Y = Y_scaled
```

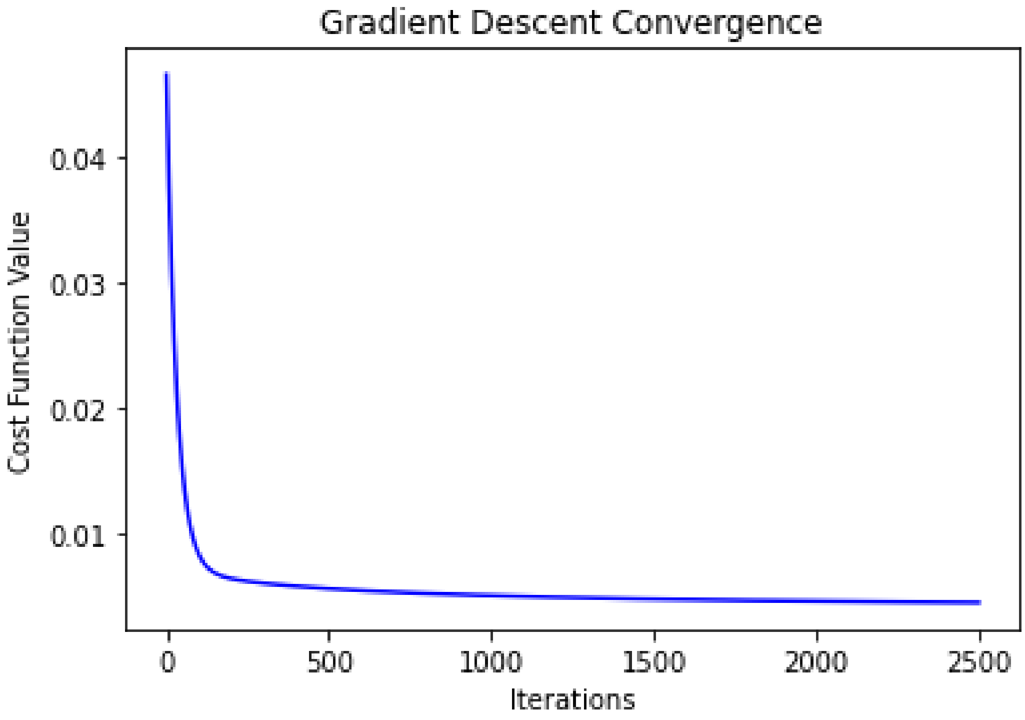
[16]:

```
# create a NumPy array 'X_T' containing the transpose of the feature matrix 'X'
X_T = np.array(X.T)

# retrieve the number of rows 'm' and the number of columns 'n' from the␣
 ↪feature matrix 'X'
m, n = X.shape

# display the values of 'm' and 'n'
print("Number of training examples (m): ", m)
print("Number of features (n): ", n)

# set the number of iterations for gradient descent
iterations = 2500

# create a counter variable 'count' and a NumPy array 'j' to store the cost␣
 ↪function values for each iteration
count = 0
j = np.zeros(shape=(iterations, 1), dtype=float)

# display the shape of the 'j' array
print("Shape of 'j' array: ", j.shape)
```

```
Number of training examples (m):  436
Number of features (n):  12
Shape of 'j' array:  (2500, 1)
```

[17]:

```
# set the initial iteration count to zero
count = 0

# create a NumPy array 'j' to store the cost function values for each iteration
j = np.zeros(shape=(iterations, 1), dtype=float)

# perform gradient descent for the specified number of iterations
while count < iterations:

    # calculate the predicted values 'h' using the current parameters 'theta'
    h = X.dot(theta)

    # calculate the cost function value 'j' using the current parameters 'theta'
    j[count] = (1/(2*m)) * np.sum((h-Y)**2)

    # calculate the gradient of the cost function with respect to 'theta'
    grad = (1/m) * X_T.dot(h-Y)
    # update the parameters 'theta' using the learning rate 'alpha' and the gradient 'grad'
    alpha = 0.01
    theta = theta - alpha * grad

    # increment the iteration count
    count += 1

# plot the cost function values over the iterations
plt.plot(j,'b-')
plt.xlabel('Iterations')
plt.ylabel('Cost Function Value')
plt.title('Gradient Descent Convergence')
plt.show()
```



[18]:

```
# extract the test set features into NumPy arrays
N_X1_t = np.array(test.area)
N_X2_t = np.array(test.bedrooms)
N_X3_t = np.array(test.bathrooms)
N_X4_t = np.array(test.stories)
N_X5_t = np.array(test.parking)
N_X6_t = np.array(test.mainroad)
N_X7_t = np.array(test.guestroom)
```

```
N_X8_t = np.array(test.hotwaterheating)
N_X9_t = np.array(test.airconditioning)
N_X10_t = np.array(test.prefarea)
N_X11_t = np.array(test.basement)
N_X0_t = np.ones(109)
```

[19]:

```
# stack the test set features into a design matrix
N_X = np.vstack([N_X0_t,N_X1_t,N_X2_t,N_X3_t,N_X4_t,N_X5_t,␣
 ↪N_X6_t,N_X7_t,N_X8_t,N_X9_t,N_X10_t,N_X11_t])
N_X_T = N_X.T
N_X = np.array(N_X_T)
N_x = scaler.fit_transform(N_X)
N_X = N_x
N_X.shape
```

[19]:

```
(109, 12)
```

[20]:

```
# initialize the parameters for the test set
N_theta = np.array([0.,0.,0.,0.,0,0.,0.,0.,0.,0.,0,0.])
N_theta = N_theta.reshape(12,1)
N_theta
```

[20]:

```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[21]:

```
# initialize the target variable for the test set
N_Y_t = np.array(test.price)
N_Y = N_Y_t
N_Y = N_Y_t.reshape(109,1)
N_y = scaler.fit_transform(N_Y)
N_Y=N_y
N_Y.shape
```

[21]:

```
(109, 1)
```

[22]:

```
N_X_T = np.array(N_X.T)
m,n = N_X.shape
```

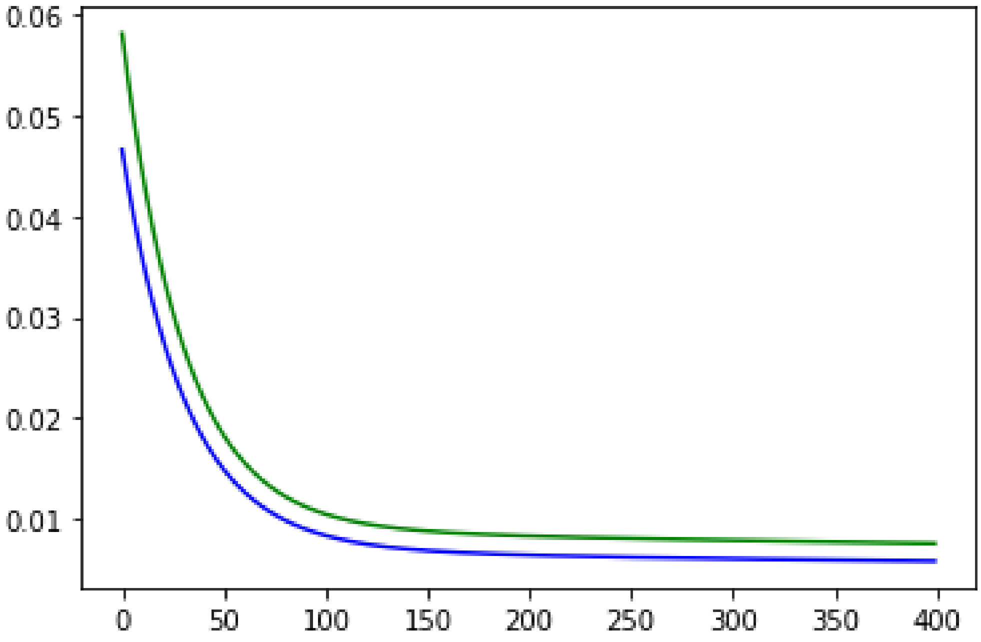
```
m,n
```

[22]:
```
(109, 12)
```

[23]:
```
iterations = 2500
count=0
N_j = np.zeros(shape=(iterations, 1), dtype=float)

while(count < iterations):

    N_h_t = N_X.dot(N_theta)
    N_h = np.array(N_h_t)


    N_j[count]= (1/(2*m))*np.sum((N_h - N_Y)**2)

    grad_t = N_X_T.dot(N_h-N_Y)
    grad = grad_t*(1/m)

    N_theta = N_theta - 0.01*(grad)

    count += 1
```

[25]:
```
plt.plot(N_j[:400],'g-')
plt.plot(j[:400],'b-')
```

[25]:
```
[<matplotlib.lines.Line2D at 0x7ffb24cc4f70>]
```



hw1-2a-normalization

February 20, 2023

```
[ ]: import numpy as np # import numpy library
     import pandas as pd # import pandas library
     import matplotlib.pyplot as plt # import matplotlib library
     from sklearn import preprocessing  # import scikit-learn library (source: https:
      ↪//scikit-learn.org/stable/index.html)
```

```
[ ]: df = pd.read_csv("/content/sample_data/Housing.csv")

     # display DataFrame
     df
```

```
[ ]:         price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
     0    13300000  7420         4          2        3      yes        no       no
     1    12250000  8960         4          4        4      yes        no       no
     2    12250000  9960         3          2        2      yes        no      yes
     3    12215000  7500         4          2        2      yes        no      yes
     4    11410000  7420         4          1        2      yes       yes      yes
     ..        …   …       …        …      …      …       …      …
     540   1820000  3000         2          1        1      yes        no      yes
     541   1767150  2400         3          1        1       no        no       no
     542   1750000  3620         2          1        1      yes        no       no
     543   1750000  2910         3          1        1       no        no       no
     544   1750000  3850         3          1        2      yes        no       no

         hotwaterheating airconditioning  parking prefarea furnishingstatus
     0                no             yes        2      yes        furnished
     1                no             yes        3       no        furnished
     2                no              no        2      yes   semi-furnished
     3                no             yes        3      yes        furnished
     4                no             yes        2       no        furnished
     ..             …             …      …      …              …
     540              no              no        2       no      unfurnished
     541              no              no        0       no   semi-furnished
     542              no              no        0       no      unfurnished
     543              no              no        0       no        furnished
     544              no              no        0       no      unfurnished
```

```
[545 rows x 13 columns]
```

```
[ ]: df = df.replace(to_replace=['yes', 'no'], value=[1, 0])
     df
```

```
[ ]:
        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0    13300000  7420         4          2        3         1          0
1    12250000  8960         4          4        4         1          0
2    12250000  9960         3          2        2         1          0
3    12215000  7500         4          2        2         1          0
4    11410000  7420         4          1        2         1          1
..        …   …       …        …      …       …         …
540   1820000  3000         2          1        1         1          0
541   1767150  2400         3          1        1         0          0
542   1750000  3620         2          1        1         1          0
543   1750000  2910         3          1        1         0          0
544   1750000  3850         3          1        2         1          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
0           0                0                1        2         1
1           0                0                1        3         0
2           1                0                0        2         1
3           1                0                1        3         1
4           1                0                1        2         0
..        …              …              …       …       …
540         1                0                0        2         0
541         0                0                0        0         0
542         0                0                0        0         0
543         0                0                0        0         0
544         0                0                0        0         0

    furnishingstatus
0          furnished
1          furnished
2     semi-furnished
3          furnished
4          furnished
..               …
540      unfurnished
541   semi-furnished
542      unfurnished
543        furnished
544      unfurnished

[545 rows x 13 columns]
```

[ ]:

```
# create a training set by randomly selecting 80% of the rows from the DataFrame
train = df.sample(frac=0.8, random_state=1)

# create a test set by dropping the rows in the training set from the DataFrame
test = df.drop(train.index)
```

[ ]:

```
test
```

[ ]:

```
        price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
2    12250000   9960         3          2        2         1          0
3    12215000   7500         4          2        2         1          0
7    10150000  16200         5          3        2         1          0
15    9100000   6000         4          1        2         1          0
22    8645000   8050         3          1        1         1          1
..        …    …       …        …      …       …        …
508   2590000   4400         2          1        1         1          0
513   2485000   4400         3          1        2         1          0
520   2450000   7700         2          1        1         1          0
537   1890000   1700         3          1        2         1          0
539   1855000   2990         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
2           1                0                0        2         1
3           1                0                1        3         1
7           0                0                0        0         0
15          1                0                0        2         0
22          1                0                1        1         0
..        …              …              …      …       …
508         0                0                0        0         0
513         0                0                0        0         0
520         0                0                0        0         0
537         0                0                0        0         0
539         0                0                0        1         0

    furnishingstatus
2     semi-furnished
3          furnished
7        unfurnished
15    semi-furnished
22         furnished
..               …
508      unfurnished
513      unfurnished
520      unfurnished
537      unfurnished
539      unfurnished
```

```
[109 rows x 13 columns]
```

```
[ ]: # select specific columns for the training set
     train = train[['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']]

     # select specific columns for the test set
     test = test[['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']]
```

```
[ ]: import sklearn.preprocessing # import scikit-learn library for data␣
      ↪preprocessing

     # create an instance of the MinMaxScaler class for scaling features to a range␣
      ↪of [0, 1]
     scaler = sklearn.preprocessing.MinMaxScaler()
```

```
[ ]: train
```

```
[ ]:        price   area  bedrooms  bathrooms  stories  parking
     62   7070000   6240         4          2        2        1
     247  4550000   8400         4          1        4        3
     142  5600000  10500         4          2        2        1
     107  6125000   6420         3          1        3        0
     483  2940000   6615         3          1        2        0
     ..       …      …        …          …        …        …
     359  3710000   3600         3          1        1        1
     36   8043000   7482         3          2        3        1
     30   8400000   7475         3          2        4        2
     20   8750000   4320         3          1        2        2
     527  2275000   1836         2          1        1        0

     [436 rows x 6 columns]
```

```
[ ]: # create a NumPy array of the 'area' column from the training set
     X1_t = np.array(train['area'])

     # display the NumPy array
     X1_t
```

```
[ ]: array([ 6240,  8400, 10500,  6420,  6615,  3600,  3240,  6600,  2700,
             5000,  2650,  4775,  4800,  3700,  7700,  7420,  4280,  6000,
             6600,  3649,  3420,  5500,  3630,  3180,  3600,  8400,  3000,
             8880,  5750,  2145,  6360,  6525,  1950,  5850,  8372,  2870,
             4990,  2684,  5200,  6321,  4960,  3480,  3210,  4950,  6840,
             4350,  5850,  4410,  2500,  3850,  3180,  3162,  3500,  4340,
             6440,  5010,  3000,  4920,  3760,  3816,  6000,  7000,  3640,
             4080,  4160,  2910,  6060,  3000,  2787,  4815,  4785,  6600,
             5300,  3600,  6000,  2176,  3000,  7420,  7020,  3480,  5960,
             3510,  6420,  6450,  6210,  4500,  3000,  3180,  5700,  3520,
             4040,  5800,  2800,  6480,  4960,  4260,  7500,  5880, 10500,
             4500,  3850,  8500,  3120,  3990,  4095,  4800, 13200,  7770,
             6100,  4075,  6550,  4100,  4370,  3180,  7350,  3510,  3640,
             5500,  8250,  6600,  8250,  2475,  3850,  4500,  3720,  4360,
            10240,  5500,  3970,  3450,  3850,  5500,  3520,  2145,  6600,
             3640,  3986,  2953,  8250,  4130,  8580,  6000,  3500,  5885,
             7680,  2430,  3150,  6450,  8100,  5500,  1650,  3040,  4079,
             2747,  4600,  2325,  7231,  3520,  2145,  3450,  3620,  4000,
             6000,  6000,  4500,  3540,  7200,  3120,  4000,  2015,  4040,
             8000,  2787,  3512,  3420,  6060,  4500,  6360,  5450,  8250,
             3960,  7410, 10360,  3630,  6020,  4100,  6254,  4500,  4560,
             6710,  3500,  8880,  3600,  7152,  6000,  4040,  4000,  4040,
             5360,  6600,  3800,  3960,  4900,  3480,  3584,  2275,  4000,
             6500, 10500,  8960,  3290,  8875,  8580,  3450,  6600,  2800,
             5640,  3745, 10269,  6100, 12090,  5880,  6750,  6000,  5320,
             4000,  4040, 15600,  3090,  3970,  5450,  4770,  4095,  6000,
             6540,  6550,  4320,  3100,  4050,  3650,  3850,  5600,  2817,
             4510,  3000,  4995, 11410,  3000,  4840,  3600,  4000,  3500,
             7800,  5300,  4840,  3000,  3480,  2970,  5828,  3800,  4040,
            10700,  7320,  5000,  6325,  2880,  4300,  3150,  4000,  9500,
             4500,  3420,  3180,  2145,  5400,  3630,  6750,  4820,  5136,
             4120,  6825,  4600,  6650,  5800,  5720,  5000,  4352,  3300,
             2160,  5900,  3000,  4500,  3350,  5400,  4600,  9800,  3630,
             2610,  9667,  3635,  4000,  3180,  3630,  6600,  2610,  4960,
             5150,  6000,  3640,  2910,  3650,  3450,  4032,  7980,  1905,
             6000,  3360,  9620,  1950, 12900,  3240,  4320,  6540,  6000,
             7440,  3760,  8100,  4880,  6000,  2000,  5200,  4050,  9166,
             7950,  5500,  2700,  6000,  6900,  3500,  5076,  5985,  4300,
             8050,  5320,  5960,  7000,  7260,  6360,  3000,  3460, 12944,
             3880,  2400,  4080,  6000,  4500,  6050,  7000,  3930,  4600,
             7155,  4100,  2400,  3460,  4632,  4200,  4640,  8800,  3000,
             6300,  7000,  7000,  6900,  3420,  3264,  2640,  3150,  4320,
             6862, 11440,  4992,  3069,  3185,  3750,  5300,  7200,  6400,
             6800,  3400,  6420,  3792,  5500,  4600,  6800,  6000,  8520,
             6480,  8150,  5948,  3185,  5830,  3410,  3000,  8400,  6350,
             8100,  4800,  2856,  3185,  3780,  3640,  6000,  6000,  4800,
             5800,  6360,  4120,  5400,  2850,  5400,  2145,  4500,  3240,
            13200,  3900,  9000,  4646,  3840,  9000,  3520,  3640,  3600,
             7482,  7475,  4320,  1836])
```

[ ]:

```
X1_t = np.array(train['area']) # 'area' column
X2_t = np.array(train['bedrooms']) # 'bedrooms' column
X3_t = np.array(train['bathrooms']) # 'bathrooms' column
X4_t = np.array(train['stories']) # 'stories' column
X5_t = np.array(train['parking']) # 'parking' column
```

```
# create a NumPy array of ones to represent the bias term
X0_t = np.ones(len(train))
```

```
[ ]: # stack the selected feature arrays vertically using np.vstack
     X = np.vstack([X0_t, X1_t, X2_t, X3_t, X4_t, X5_t])

     # transpose the stacked array to make it a 6 x 436 matrix
     X = X.T

     # convert the stacked array to a NumPy array
     X = np.array(X)

     # display the NumPy array
     X
```

```
[ ]: array([[1.000e+00, 6.240e+03, 4.000e+00, 2.000e+00, 2.000e+00, 1.000e+00],
            [1.000e+00, 8.400e+03, 4.000e+00, 1.000e+00, 4.000e+00, 3.000e+00],
            [1.000e+00, 1.050e+04, 4.000e+00, 2.000e+00, 2.000e+00, 1.000e+00],
            …,
            [1.000e+00, 7.475e+03, 3.000e+00, 2.000e+00, 4.000e+00, 2.000e+00],
            [1.000e+00, 4.320e+03, 3.000e+00, 1.000e+00, 2.000e+00, 2.000e+00],
            [1.000e+00, 1.836e+03, 2.000e+00, 1.000e+00, 1.000e+00, 0.000e+00]])
```

```
[ ]: # scale the feature matrix using the fit_transform() method of the scaler object
     X_scaled = scaler.fit_transform(X)

     # assign the scaled feature matrix to the original variable name 'X'
     X = X_scaled
```

```
[ ]: # create a 1D NumPy array of zeros with length 6
     theta = np.zeros(6)

     # reshape the 1D array to a column vector using np.reshape
     theta = np.reshape(theta, (6,1))

     # display the column vector
     theta
```

```
[ ]: array([[0.],
            [0.],
            [0.],
            [0.],
            [0.],
            [0.]])
```

```
[ ]: # create a 1D NumPy array 'Y_t' from the 'price' column of the training set␣
       ↪'train'
```

```
Y_t = np.array(train.price)

# create a copy of 'Y_t' to prevent changing the original data
Y = Y_t.copy()

# reshape 'Y' to a column vector using np.reshape
Y = np.reshape(Y, (436,1))

# create an instance of the MinMaxScaler class for scaling the target variable␣
 ↪to a range of [0, 1]
scaler = sklearn.preprocessing.MinMaxScaler()

# scale the target variable using the fit_transform() method of the scaler␣
 ↪object
Y_scaled = scaler.fit_transform(Y)

# assign the scaled target variable to the original variable name 'Y'
Y = Y_scaled
```

[ ]:

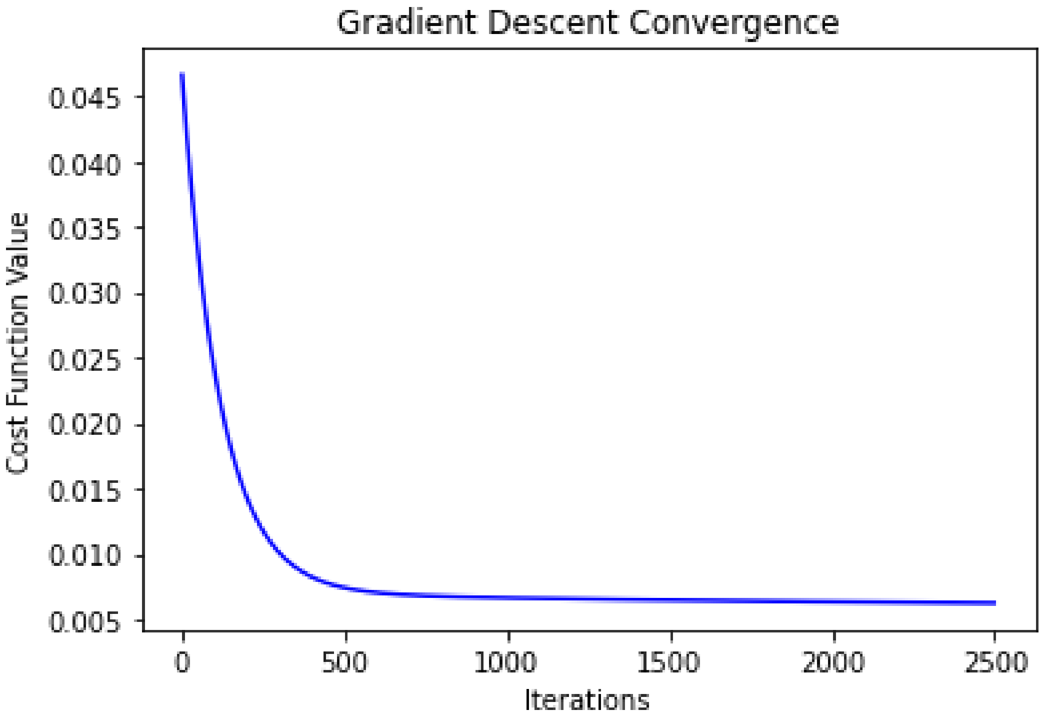
```
# create a NumPy array 'X_T' containing the transpose of the feature matrix 'X'
X_T = np.array(X.T)

# retrieve the number of rows 'm' and the number of columns 'n' from the␣
 ↪feature matrix 'X'
m, n = X.shape

# display the values of 'm' and 'n'
print("Number of training examples (m): ", m)
print("Number of features (n): ", n)

# set the number of iterations for gradient descent
iterations = 2500

# create a counter variable 'count' and a NumPy array 'j' to store the cost␣
 ↪function values for each iteration
count = 0
j = np.zeros(shape=(iterations, 1), dtype=float)

# display the shape of the 'j' array
print("Shape of 'j' array: ", j.shape)
```

```
Number of training examples (m):  436
Number of features (n):  6
Shape of 'j' array:  (2500, 1)
```

[ ]:

```
# set the initial iteration count to zero
count = 0
```

```
# create a NumPy array 'j' to store the cost function values for each iteration
j = np.zeros(shape=(iterations, 1), dtype=float)

# perform gradient descent for the specified number of iterations
while count < iterations:

    # calculate the predicted values 'h' using the current parameters 'theta'
    h = X.dot(theta)

    # calculate the cost function value 'j' using the current parameters 'theta'
    j[count] = (1/(2*m)) * np.sum((h-Y)**2)

    # calculate the gradient of the cost function with respect to 'theta'
    grad = (1/m) * X_T.dot(h-Y)

    # update the parameters 'theta' using the learning rate 'alpha' and the gradient 'grad'
    alpha = 0.01
    theta = theta - alpha * grad

    # increment the iteration count
    count += 1

# plot the cost function values over the iterations
plt.plot(j,'b-')
plt.xlabel('Iterations')
plt.ylabel('Cost Function Value')
plt.title('Gradient Descent Convergence')
plt.show()
```



[ ]:
```
# extract the test set features into NumPy arrays
N_X1_t = np.array(test.area)
N_X2_t = np.array(test.bedrooms)
N_X3_t = np.array(test.bathrooms)
N_X4_t = np.array(test.stories)
N_X5_t = np.array(test.parking)
N_X0_t = np.ones(109)
```

[ ]:
```
# stack the test set features into a design matrix
N_X = np.vstack([N_X0_t,N_X1_t,N_X2_t,N_X3_t,N_X4_t,N_X5_t])
N_X_T = N_X.T
N_X = np.array(N_X_T)
N_x = scaler.fit_transform(N_X)
N_X = N_x
N_X.shape
```

[ ]:
```
(109, 6)
```

[ ]:
```
N_theta = np.array([0.,0.,0.,0.,0,0.])
N_theta = N_theta.reshape(6,1)
N_theta
```

```
[ ]: array([[0.],
            [0.],
            [0.],
            [0.],
            [0.],
            [0.]])
```

```
[ ]: N_Y_t = np.array(test.price)
     N_Y = N_Y_t
     N_Y = N_Y_t.reshape(109,1)
     N_y = scaler.fit_transform(N_Y)
     N_Y=N_y
     N_Y.shape
```

```
[ ]: (109, 1)
```

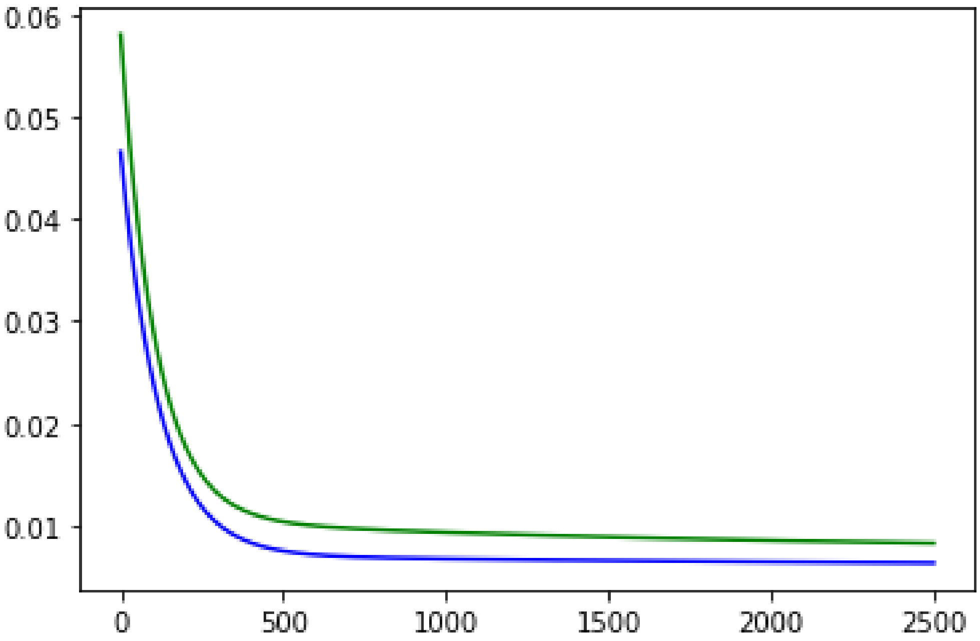
```
[ ]: N_X_T = np.array(N_X.T)
     m,n = N_X.shape
     m,n
```

```
[ ]: (109, 6)
```

```
[ ]: iterations = 2500
     count=0
     N_j = np.zeros(shape=(iterations, 1), dtype=float)

     while(count < iterations):

         N_h_t = N_X.dot(N_theta)
         N_h = np.array(N_h_t)


         N_j[count]= (1/(2*m))*np.sum((N_h - N_Y)**2)

         grad_t = N_X_T.dot(N_h-N_Y)
         grad = grad_t*(1/m)

         N_theta = N_theta - 0.01*(grad)

         count += 1
```

```
[ ]: plt.plot(N_j,'g-')
     plt.plot(j,'b-')
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7fdb15afbe20>]
```


0.06
0.05
0.04
0.03
0.02
0.01
0
500
1000
1500
2000
2500

hw1-2a-standardization

February 20, 2023

[ ]:

```
import numpy as np # import numpy library
import pandas as pd # import pandas library
import matplotlib.pyplot as plt # import matplotlib library
from sklearn import preprocessing
```

[ ]:

```
df = pd.read_csv("/content/sample_data/Housing.csv")

# display DataFrame
df
```

[ ]:

```
        price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0    13300000  7420         4          2        3      yes        no       no
1    12250000  8960         4          4        4      yes        no       no
2    12250000  9960         3          2        2      yes        no      yes
3    12215000  7500         4          2        2      yes        no      yes
4    11410000  7420         4          1        2      yes       yes      yes
..        …     …         …          …        …        …         …
540   1820000  3000         2          1        1      yes        no      yes
541   1767150  2400         3          1        1       no        no       no
542   1750000  3620         2          1        1      yes        no       no
543   1750000  2910         3          1        1       no        no       no
544   1750000  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
0                no             yes        2      yes        furnished
1                no             yes        3       no        furnished
2                no              no        2      yes   semi-furnished
3                no             yes        3      yes        furnished
4                no             yes        2       no        furnished
..              …               …       …        …              …
540              no              no        2       no      unfurnished
541              no              no        0       no   semi-furnished
542              no              no        0       no      unfurnished
543              no              no        0       no        furnished
544              no              no        0       no      unfurnished

[545 rows x 13 columns]
```

[ ]:
```
# create a training set by randomly selecting 80% of the rows from the DataFrame
train = df.sample(frac=0.8, random_state=1)

# create a test set by dropping the rows in the training set from the DataFrame
test = df.drop(train.index)
```

[ ]:
```
train
```

[ ]:
```
       price   area  bedrooms  bathrooms  stories mainroad guestroom basement  \
62   7070000   6240         4          2        2      yes        no       no
247  4550000   8400         4          1        4      yes        no       no
142  5600000  10500         4          2        2      yes        no       no
107  6125000   6420         3          1        3      yes        no      yes
483  2940000   6615         3          1        2      yes        no       no
..       …      …         …          …        …      …         …
359  3710000   3600         3          1        1      yes        no       no
36   8043000   7482         3          2        3      yes        no       no
30   8400000   7475         3          2        4      yes        no       no
20   8750000   4320         3          1        2      yes        no      yes
527  2275000   1836         2          1        1       no        no      yes

    hotwaterheating airconditioning  parking prefarea furnishingstatus
62               no             yes        1       no        furnished
247              no              no        3       no      unfurnished
142              no              no        1       no   semi-furnished
107              no              no        0      yes      unfurnished
483              no              no        0       no   semi-furnished
..              …             …      …      …              …
359              no              no        1       no      unfurnished
36              yes              no        1      yes        furnished
30               no             yes        2       no      unfurnished
20              yes              no        2       no   semi-furnished
527              no              no        0       no   semi-furnished

[436 rows x 13 columns]
```

[ ]:
```
test
```

[ ]:
```
        price   area  bedrooms  bathrooms  stories mainroad guestroom  \
2    12250000   9960         3          2        2      yes        no
3    12215000   7500         4          2        2      yes        no
7    10150000  16200         5          3        2      yes        no
15    9100000   6000         4          1        2      yes        no
22    8645000   8050         3          1        1      yes       yes
..        …      …         …          …        …      …         …
508   2590000   4400         2          1        1      yes        no
513   2485000   4400         3          1        2      yes        no
```

```
520   2450000   7700         2          1        1      yes        no
537   1890000   1700         3          1        2      yes        no
539   1855000   2990         2          1        1       no        no

    basement hotwaterheating airconditioning  parking prefarea  \
2        yes              no              no        2      yes
3        yes              no             yes        3      yes
7         no              no              no        0       no
15       yes              no              no        2       no
22       yes              no             yes        1       no
..       …             …             …      …      …
508       no              no              no        0       no
513       no              no              no        0       no
520       no              no              no        0       no
537       no              no              no        0       no
539       no              no              no        1       no

    furnishingstatus
2     semi-furnished
3          furnished
7        unfurnished
15    semi-furnished
22         furnished
..                 …
508      unfurnished
513      unfurnished
520      unfurnished
537      unfurnished
539      unfurnished

[109 rows x 13 columns]
```

[ ]:

```python
# select specific columns for the training set
train = train[['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']]

# select specific columns for the test set
test = test[['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']]
```

[ ]:

```python
scaler = preprocessing.StandardScaler()
```

[ ]:

```python
# create a NumPy array of the 'area' column from the training set
X1_t = np.array(train['area'])

# display the NumPy array
X1_t
```

```
[ ]: array([ 6240,  8400, 10500,  6420,  6615,  3600,  3240,  6600,  2700,
             5000,  2650,  4775,  4800,  3700,  7700,  7420,  4280,  6000,
             6600,  3649,  3420,  5500,  3630,  3180,  3600,  8400,  3000,
             8880,  5750,  2145,  6360,  6525,  1950,  5850,  8372,  2870,
             4990,  2684,  5200,  6321,  4960,  3480,  3210,  4950,  6840,
             4350,  5850,  4410,  2500,  3850,  3180,  3162,  3500,  4340,
             6440,  5010,  3000,  4920,  3760,  3816,  6000,  7000,  3640,
             4080,  4160,  2910,  6060,  3000,  2787,  4815,  4785,  6600,
             5300,  3600,  6000,  2176,  3000,  7420,  7020,  3480,  5960,
             3510,  6420,  6450,  6210,  4500,  3000,  3180,  5700,  3520,
             4040,  5800,  2800,  6480,  4960,  4260,  7500,  5880, 10500,
             4500,  3850,  8500,  3120,  3990,  4095,  4800, 13200,  7770,
             6100,  4075,  6550,  4100,  4370,  3180,  7350,  3510,  3640,
             5500,  8250,  6600,  8250,  2475,  3850,  4500,  3720,  4360,
            10240,  5500,  3970,  3450,  3850,  5500,  3520,  2145,  6600,
             3640,  3986,  2953,  8250,  4130,  8580,  6000,  3500,  5885,
             7680,  2430,  3150,  6450,  8100,  5500,  1650,  3040,  4079,
             2747,  4600,  2325,  7231,  3520,  2145,  3450,  3620,  4000,
             6000,  6000,  4500,  3540,  7200,  3120,  4000,  2015,  4040,
             8000,  2787,  3512,  3420,  6060,  4500,  6360,  5450,  8250,
             3960,  7410, 10360,  3630,  6020,  4100,  6254,  4500,  4560,
             6710,  3500,  8880,  3600,  7152,  6000,  4040,  4000,  4040,
             5360,  6600,  3800,  3960,  4900,  3480,  3584,  2275,  4000,
             6500, 10500,  8960,  3290,  8875,  8580,  3450,  6600,  2800,
             5640,  3745, 10269,  6100, 12090,  5880,  6750,  6000,  5320,
             4000,  4040, 15600,  3090,  3970,  5450,  4770,  4095,  6000,
             6540,  6550,  4320,  3100,  4050,  3650,  3850,  5600,  2817,
             4510,  3000,  4995, 11410,  3000,  4840,  3600,  4000,  3500,
             7800,  5300,  4840,  3000,  3480,  2970,  5828,  3800,  4040,
            10700,  7320,  5000,  6325,  2880,  4300,  3150,  4000,  9500,
             4500,  3420,  3180,  2145,  5400,  3630,  6750,  4820,  5136,
             4120,  6825,  4600,  6650,  5800,  5720,  5000,  4352,  3300,
             2160,  5900,  3000,  4500,  3350,  5400,  4600,  9800,  3630,
             2610,  9667,  3635,  4000,  3180,  3630,  6600,  2610,  4960,
             5150,  6000,  3640,  2910,  3650,  3450,  4032,  7980,  1905,
             6000,  3360,  9620,  1950, 12900,  3240,  4320,  6540,  6000,
             7440,  3760,  8100,  4880,  6000,  2000,  5200,  4050,  9166,
             7950,  5500,  2700,  6000,  6900,  3500,  5076,  5985,  4300,
             8050,  5320,  5960,  7000,  7260,  6360,  3000,  3460, 12944,
             3880,  2400,  4080,  6000,  4500,  6050,  7000,  3930,  4600,
             7155,  4100,  2400,  3460,  4632,  4200,  4640,  8800,  3000,
             6300,  7000,  7000,  6900,  3420,  3264,  2640,  3150,  4320,
             6862, 11440,  4992,  3069,  3185,  3750,  5300,  7200,  6400,
             6800,  3400,  6420,  3792,  5500,  4600,  6800,  6000,  8520,
             6480,  8150,  5948,  3185,  5830,  3410,  3000,  8400,  6350,
             8100,  4800,  2856,  3185,  3780,  3640,  6000,  6000,  4800,
             5800,  6360,  4120,  5400,  2850,  5400,  2145,  4500,  3240,
            13200,  3900,  9000,  4646,  3840,  9000,  3520,  3640,  3600,
             7482,  7475,  4320,  1836])
```

```
[ ]: X1_t = np.array(train['area']) # 'area' column
     X2_t = np.array(train['bedrooms']) # 'bedrooms' column
     X3_t = np.array(train['bathrooms']) # 'bathrooms' column
     X4_t = np.array(train['stories']) # 'stories' column
     X5_t = np.array(train['parking']) # 'parking' column

     # create a NumPy array of ones to represent the bias term
     X0_t = np.ones(len(train))
```

```
[ ]: # stack the selected feature arrays vertically using np.vstack
     X = np.vstack([X0_t, X1_t, X2_t, X3_t, X4_t, X5_t])

     # transpose the stacked array to make it a 6 x 436 matrix
     X = X.T

     # convert the stacked array to a NumPy array
     X = np.array(X)

     # display the NumPy array
     X
```

```
[ ]: array([[1.000e+00, 6.240e+03, 4.000e+00, 2.000e+00, 2.000e+00, 1.000e+00],
            [1.000e+00, 8.400e+03, 4.000e+00, 1.000e+00, 4.000e+00, 3.000e+00],
            [1.000e+00, 1.050e+04, 4.000e+00, 2.000e+00, 2.000e+00, 1.000e+00],
            …,
            [1.000e+00, 7.475e+03, 3.000e+00, 2.000e+00, 4.000e+00, 2.000e+00],
            [1.000e+00, 4.320e+03, 3.000e+00, 1.000e+00, 2.000e+00, 2.000e+00],
            [1.000e+00, 1.836e+03, 2.000e+00, 1.000e+00, 1.000e+00, 0.000e+00]])
```

```
[ ]: x = scaler.fit_transform(X)
     X = x
```

```
[ ]: # create a 1D NumPy array of zeros with length 6
     theta = np.zeros(6)

     # reshape the 1D array to a column vector using np.reshape
     theta = np.reshape(theta, (6,1))

     # display the column vector
     theta
```

```
[ ]: array([[0.],
            [0.],
            [0.],
       [0.],
       [0.],
       [0.]])
```

[ ]:

```
# create a 1D NumPy array 'Y_t' from the 'price' column of the training set␣
 ↪'train'
#h = np.matmul(X,theta)
Y_t = np.array(train.price)
Y = Y_t
Y = Y_t.reshape(436,1)
y = scaler.fit_transform(Y)
Y=y
```

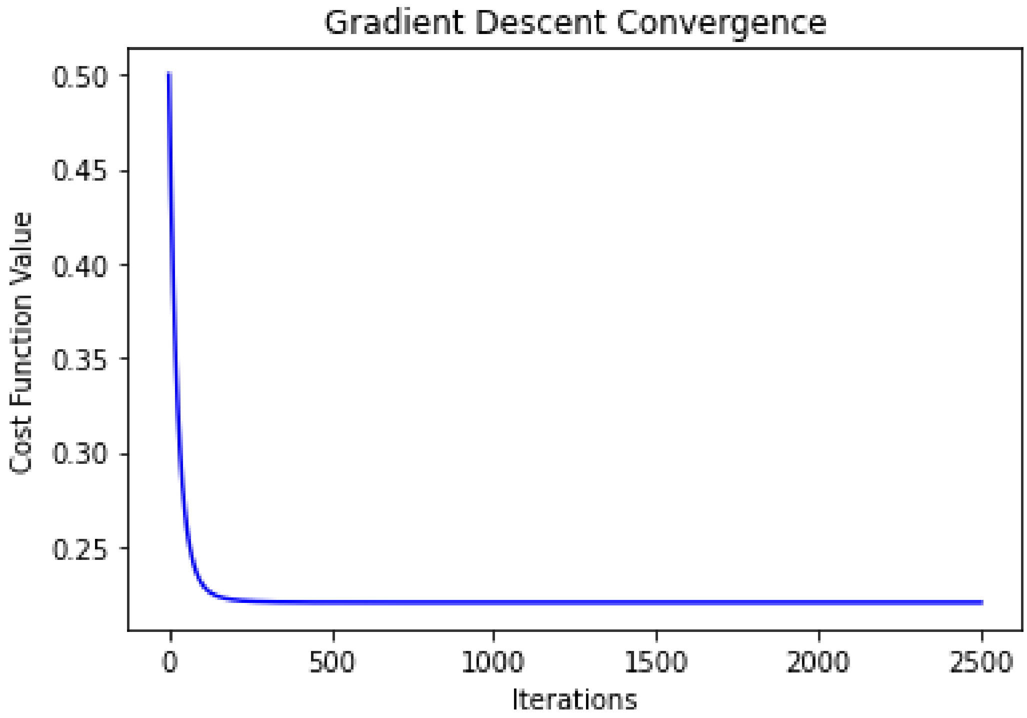
[ ]:

```
# create a NumPy array 'X_T' containing the transpose of the feature matrix 'X'
X_T = np.array(X.T)

# retrieve the number of rows 'm' and the number of columns 'n' from the␣
 ↪feature matrix 'X'
m, n = X.shape

# display the values of 'm' and 'n'
print("Number of training examples (m): ", m)
print("Number of features (n): ", n)

# set the number of iterations for gradient descent
iterations = 2500

# create a counter variable 'count' and a NumPy array 'j' to store the cost␣
 ↪function values for each iteration
count = 0
j = np.zeros(shape=(iterations, 1), dtype=float)

# display the shape of the 'j' array
print("Shape of 'j' array: ", j.shape)
```

```
Number of training examples (m):  436
Number of features (n):  6
Shape of 'j' array:  (2500, 1)
```

[ ]:

```
# set the initial iteration count to zero
count = 0

# create a NumPy array 'j' to store the cost function values for each iteration
j = np.zeros(shape=(iterations, 1), dtype=float)

# perform gradient descent for the specified number of iterations
while count < iterations:
    # calculate the predicted values 'h' using the current parameters 'theta'
    h = X.dot(theta)

    # calculate the cost function value 'j' using the current parameters 'theta'
    j[count] = (1/(2*m)) * np.sum((h-Y)**2)

    # calculate the gradient of the cost function with respect to 'theta'
    grad = (1/m) * X_T.dot(h-Y)

    # update the parameters 'theta' using the learning rate 'alpha' and the gradient 'grad'
    alpha = 0.01
    theta = theta - alpha * grad

    # increment the iteration count
    count += 1

# plot the cost function values over the iterations
plt.plot(j,'b-')
plt.xlabel('Iterations')
plt.ylabel('Cost Function Value')
plt.title('Gradient Descent Convergence')
plt.show()
```



[ ]:

```
# extract the test set features into NumPy arrays
N_X1_t = np.array(test.area)
N_X2_t = np.array(test.bedrooms)
N_X3_t = np.array(test.bathrooms)
N_X4_t = np.array(test.stories)
N_X5_t = np.array(test.parking)
N_X0_t = np.ones(109)
```

[ ]:

```
# stack the test set features into a design matrix
N_X = np.vstack([N_X0_t,N_X1_t,N_X2_t,N_X3_t,N_X4_t,N_X5_t])
N_X_T = N_X.T
N_X = np.array(N_X_T)
N_x = scaler.fit_transform(N_X)
N_X = N_x
N_X.shape
```

[ ]:

```
(109, 6)
```

[ ]:

```
N_theta = np.array([0.,0.,0.,0.,0,0.])
N_theta = N_theta.reshape(6,1)
N_theta
```

[ ]:

```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[ ]:

```
N_Y_t = np.array(test.price)
N_Y = N_Y_t
N_Y = N_Y_t.reshape(109,1)
N_y = scaler.fit_transform(N_Y)
N_Y=N_y
N_Y.shape
```

[ ]:

```
(109, 1)
```

[ ]:

```
N_X_T = np.array(N_X.T)
m,n = N_X.shape
m,n
```

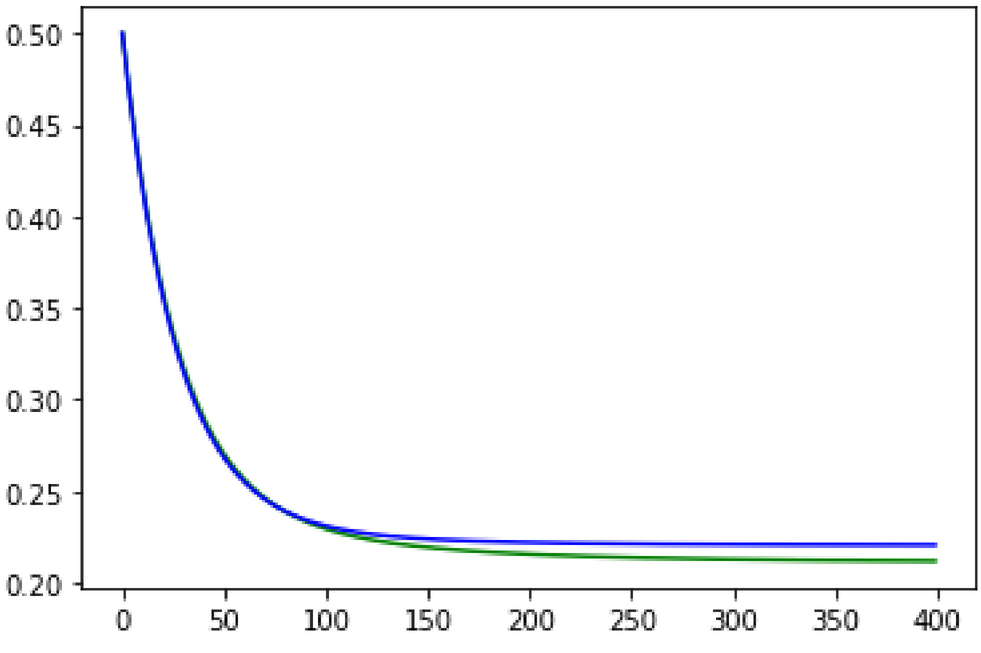
[ ]:

```
(109, 6)
```

```
[ ]: iterations = 2500
     count=0
     N_j = np.zeros(shape=(iterations, 1), dtype=float)

     while(count < iterations):

         N_h_t = N_X.dot(N_theta)
         N_h = np.array(N_h_t)


         N_j[count]= (1/(2*m))*np.sum((N_h - N_Y)**2)

         grad_t = N_X_T.dot(N_h-N_Y)
         grad = grad_t*(1/m)

         N_theta = N_theta - 0.01*(grad)

         count += 1
```

```
[ ]: plt.plot(N_j[:400],'g-')
     plt.plot(j[:400],'b-')
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7f9ecbdcf5b0>]
```



hw1-2b-normalization

February 20, 2023

```
[5]: import numpy as np # import numpy library
     import pandas as pd # import pandas library
     import matplotlib.pyplot as plt # import matplotlib library
     from sklearn import preprocessing  # import scikit-learn library (source: https:
      ↪//scikit-learn.org/stable/index.html)
```

```
[ ]: df = pd.read_csv("/content/sample_data/Housing.csv")

     # display DataFrame
     df
```

```
[8]: df = df.replace(to_replace=['yes', 'no'], value=[1, 0])
     df
```

```
[8]:         price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
     0    13300000  7420         4          2        3         1          0
     1    12250000  8960         4          4        4         1          0
     2    12250000  9960         3          2        2         1          0
     3    12215000  7500         4          2        2         1          0
     4    11410000  7420         4          1        2         1          1
     ..          …     …         …          …        …         …
     540   1820000  3000         2          1        1         1          0
     541   1767150  2400         3          1        1         0          0
     542   1750000  3620         2          1        1         1          0
     543   1750000  2910         3          1        1         0          0
     544   1750000  3850         3          1        2         1          0

          basement  hotwaterheating  airconditioning  parking  prefarea  \
     0           0                0                1        2         1
     1           0                0                1        3         0
     2           1                0                0        2         1
     3           1                0                1        3         1
     4           1                0                1        2         0
     ..          …                …                …        …         …
     540         1                0                0        2         0
     541         0                0                0        0         0
     542         0                0                0        0         0
```

```
543         0                0                0        0         0
544         0                0                0        0         0

    furnishingstatus
0          furnished
1          furnished
2     semi-furnished
3          furnished
4          furnished
..               …
540      unfurnished
541   semi-furnished
542      unfurnished
543        furnished
544      unfurnished

[545 rows x 13 columns]
```

[9]:

```
# create a training set by randomly selecting 80% of the rows from the DataFrame
train = df.sample(frac=0.8, random_state=1)

# create a test set by dropping the rows in the training set from the DataFrame
test = df.drop(train.index)
```

[10]:

```
train
```

[10]:

```
       price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
62   7070000   6240         4          2        2         1          0
247  4550000   8400         4          1        4         1          0
142  5600000  10500         4          2        2         1          0
107  6125000   6420         3          1        3         1          0
483  2940000   6615         3          1        2         1          0
..       …    …       …        …      …        …        …
359  3710000   3600         3          1        1         1          0
36   8043000   7482         3          2        3         1          0
30   8400000   7475         3          2        4         1          0
20   8750000   4320         3          1        2         1          0
527  2275000   1836         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
62          0                0                1        1         0
247         0                0                0        3         0
142         0                0                0        1         0
107         1                0                0        0         1
483         0                0                0        0         0
..        …              …               …      …       …
359         0                0                0        1         0
```

```
36          0                0                0        1         1
30          0                0                1        2         0
20          1                1                0        2         0
527         1                0                0        0         0

    furnishingstatus
62         furnished
247      unfurnished
142   semi-furnished
107      unfurnished
483   semi-furnished
..               …
359      unfurnished
36         furnished
30       unfurnished
20    semi-furnished
527   semi-furnished

[436 rows x 13 columns]
```

```
[11]: test
```

```
[11]:
        price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
2    12250000   9960         3          2        2         1          0
3    12215000   7500         4          2        2         1          0
7    10150000  16200         5          3        2         1          0
15    9100000   6000         4          1        2         1          0
22    8645000   8050         3          1        1         1          1
..        …     …      …        …     …        …        …
508   2590000   4400         2          1        1         1          0
513   2485000   4400         3          1        2         1          0
520   2450000   7700         2          1        1         1          0
537   1890000   1700         3          1        2         1          0
539   1855000   2990         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
2           1                0                0        2         1
3           1                0                1        3         1
7           0                0                0        0         0
15          1                0                0        2         0
22          1                0                1        1         0
..        …              …              …      …       …
508         0                0                0        0         0
513         0                0                0        0         0
520         0                0                0        0         0
537         0                0                0        0         0
539         0                0                0        1         0
```

```
    furnishingstatus
2     semi-furnished
3          furnished
7        unfurnished
15    semi-furnished
22         furnished
..               …
508      unfurnished
513      unfurnished
520      unfurnished
537      unfurnished
539      unfurnished

[109 rows x 13 columns]
```

[12]:

```
# select specific columns for the training set
train = train[['price','area','bedrooms','bathrooms','stories','parking',␣
 ↪'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',␣
 ↪'prefarea']]
# select specific columns for the test set
test = test[['price','area','bedrooms','bathrooms','stories','parking',␣
 ↪'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',␣
 ↪'prefarea']]
```

[13]:

```
import sklearn.preprocessing # import scikit-learn library for data␣
 ↪preprocessing

# create an instance of the MinMaxScaler class for scaling features to a range␣
 ↪of [0, 1]
scaler = sklearn.preprocessing.MinMaxScaler()
```

[14]:

```
train
```

[14]:

```
       price   area  bedrooms  bathrooms  stories  parking  mainroad  \
62   7070000   6240         4          2        2        1         1
247  4550000   8400         4          1        4        3         1
142  5600000  10500         4          2        2        1         1
107  6125000   6420         3          1        3        0         1
483  2940000   6615         3          1        2        0         1
..       …      …       …         …       …       …       …
359  3710000   3600         3          1        1        1         1
36   8043000   7482         3          2        3        1         1
30   8400000   7475         3          2        4        2         1
20   8750000   4320         3          1        2        2         1
527  2275000   1836         2          1        1        0         0
```

```
     guestroom  basement  hotwaterheating  airconditioning  prefarea
62           0         0                0                1         0
247          0         0                0                0         0
142          0         0                0                0         0
107          0         1                0                0         1
483          0         0                0                0         0
..         …       …             …              …       …
359          0         0                0                0         0
36           0         0                1                0         1
30           0         0                0                1         0
20           0         1                1                0         0
527          0         1                0                0         0

[436 rows x 12 columns]
```

[15]:

```
# create a NumPy array of the 'area' column from the training set
X1_t = np.array(train['area'])

# display the NumPy array
X1_t
```

[15]:

```
array([ 6240,  8400, 10500,  6420,  6615,  3600,  3240,  6600,  2700,
        5000,  2650,  4775,  4800,  3700,  7700,  7420,  4280,  6000,
        6600,  3649,  3420,  5500,  3630,  3180,  3600,  8400,  3000,
        8880,  5750,  2145,  6360,  6525,  1950,  5850,  8372,  2870,
        4990,  2684,  5200,  6321,  4960,  3480,  3210,  4950,  6840,
        4350,  5850,  4410,  2500,  3850,  3180,  3162,  3500,  4340,
        6440,  5010,  3000,  4920,  3760,  3816,  6000,  7000,  3640,
        4080,  4160,  2910,  6060,  3000,  2787,  4815,  4785,  6600,
        5300,  3600,  6000,  2176,  3000,  7420,  7020,  3480,  5960,
        3510,  6420,  6450,  6210,  4500,  3000,  3180,  5700,  3520,
        4040,  5800,  2800,  6480,  4960,  4260,  7500,  5880, 10500,
        4500,  3850,  8500,  3120,  3990,  4095,  4800, 13200,  7770,
        6100,  4075,  6550,  4100,  4370,  3180,  7350,  3510,  3640,
        5500,  8250,  6600,  8250,  2475,  3850,  4500,  3720,  4360,
       10240,  5500,  3970,  3450,  3850,  5500,  3520,  2145,  6600,
        3640,  3986,  2953,  8250,  4130,  8580,  6000,  3500,  5885,
        7680,  2430,  3150,  6450,  8100,  5500,  1650,  3040,  4079,
        2747,  4600,  2325,  7231,  3520,  2145,  3450,  3620,  4000,
        6000,  6000,  4500,  3540,  7200,  3120,  4000,  2015,  4040,
        8000,  2787,  3512,  3420,  6060,  4500,  6360,  5450,  8250,
        3960,  7410, 10360,  3630,  6020,  4100,  6254,  4500,  4560,
        6710,  3500,  8880,  3600,  7152,  6000,  4040,  4000,  4040,
        5360,  6600,  3800,  3960,  4900,  3480,  3584,  2275,  4000,
        6500, 10500,  8960,  3290,  8875,  8580,  3450,  6600,  2800,
        5640,  3745, 10269,  6100, 12090,  5880,  6750,  6000,  5320,
        4000,  4040, 15600,  3090,  3970,  5450,  4770,  4095,  6000,
        6540,  6550,  4320,  3100,  4050,  3650,  3850,  5600,  2817,
        4510,  3000,  4995, 11410,  3000,  4840,  3600,  4000,  3500,
        7800,  5300,  4840,  3000,  3480,  2970,  5828,  3800,  4040,
       10700,  7320,  5000,  6325,  2880,  4300,  3150,  4000,  9500,
        4500,  3420,  3180,  2145,  5400,  3630,  6750,  4820,  5136,
        4120,  6825,  4600,  6650,  5800,  5720,  5000,  4352,  3300,
        2160,  5900,  3000,  4500,  3350,  5400,  4600,  9800,  3630,
        2610,  9667,  3635,  4000,  3180,  3630,  6600,  2610,  4960,
        5150,  6000,  3640,  2910,  3650,  3450,  4032,  7980,  1905,
        6000,  3360,  9620,  1950, 12900,  3240,  4320,  6540,  6000,
        7440,  3760,  8100,  4880,  6000,  2000,  5200,  4050,  9166,
        7950,  5500,  2700,  6000,  6900,  3500,  5076,  5985,  4300,
        8050,  5320,  5960,  7000,  7260,  6360,  3000,  3460, 12944,
        3880,  2400,  4080,  6000,  4500,  6050,  7000,  3930,  4600,
        7155,  4100,  2400,  3460,  4632,  4200,  4640,  8800,  3000,
        6300,  7000,  7000,  6900,  3420,  3264,  2640,  3150,  4320,
        6862, 11440,  4992,  3069,  3185,  3750,  5300,  7200,  6400,
        6800,  3400,  6420,  3792,  5500,  4600,  6800,  6000,  8520,
        6480,  8150,  5948,  3185,  5830,  3410,  3000,  8400,  6350,
        8100,  4800,  2856,  3185,  3780,  3640,  6000,  6000,  4800,
        5800,  6360,  4120,  5400,  2850,  5400,  2145,  4500,  3240,
       13200,  3900,  9000,  4646,  3840,  9000,  3520,  3640,  3600,
        7482,  7475,  4320,  1836])
```

[16]:

```python
X2_t = np.array(train.bedrooms)
X3_t = np.array(train.bathrooms)
X4_t = np.array(train.stories)
X5_t = np.array(train.parking)
X6_t = np.array(train.mainroad)
X7_t = np.array(train.guestroom)
X8_t = np.array(train.hotwaterheating)
X9_t = np.array(train.airconditioning)
X10_t = np.array(train.prefarea)
X11_t = np.array(train.basement)
X0_t= np.ones(436)
```

[17]:

```python
# stack the selected feature arrays vertically using np.vstack
X = np.vstack([X0_t, X1_t, X2_t, X3_t, X4_t, X5_t,␣
 ↪X6_t,X7_t,X8_t,X9_t,X10_t,X11_t])

# transpose the stacked array to make it a 6 x 436 matrix
X = X.T

# convert the stacked array to a NumPy array
X = np.array(X)

# display the NumPy array
```

```
X
```

[17]:
```
array([[1.000e+00, 6.240e+03, 4.000e+00, …, 1.000e+00, 0.000e+00,
         0.000e+00],
        [1.000e+00, 8.400e+03, 4.000e+00, …, 0.000e+00, 0.000e+00,
         0.000e+00],
        [1.000e+00, 1.050e+04, 4.000e+00, …, 0.000e+00, 0.000e+00,
         0.000e+00],
        …,
        [1.000e+00, 7.475e+03, 3.000e+00, …, 1.000e+00, 0.000e+00,
         0.000e+00],
        [1.000e+00, 4.320e+03, 3.000e+00, …, 0.000e+00, 0.000e+00,
         1.000e+00],
        [1.000e+00, 1.836e+03, 2.000e+00, …, 0.000e+00, 0.000e+00,
         1.000e+00]])
```

[18]:
```
# scale the feature matrix using the fit_transform() method of the scaler object
X_scaled = scaler.fit_transform(X)

# assign the scaled feature matrix to the original variable name 'X'
X = X_scaled
```

[19]:
```
# create a 1D NumPy array of zeros with length 12
theta = np.zeros(12)

# reshape the 1D array to a column vector using np.reshape
theta = np.reshape(theta, (12,1))

# display the column vector
theta
```

[19]:
```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[20]:
```
# create a 1D NumPy array 'Y_t' from the 'price' column of the training set␣
 ↪'train'
Y_t = np.array(train.price)
```

```
# create a copy of 'Y_t' to prevent changing the original data
Y = Y_t.copy()

# reshape 'Y' to a column vector using np.reshape
Y = np.reshape(Y, (436,1))

# create an instance of the MinMaxScaler class for scaling the target variable␣
 ↪to a range of [0, 1]
scaler = sklearn.preprocessing.MinMaxScaler()

# scale the target variable using the fit_transform() method of the scaler␣
 ↪object
Y_scaled = scaler.fit_transform(Y)

# assign the scaled target variable to the original variable name 'Y'
Y = Y_scaled
```

[21]:

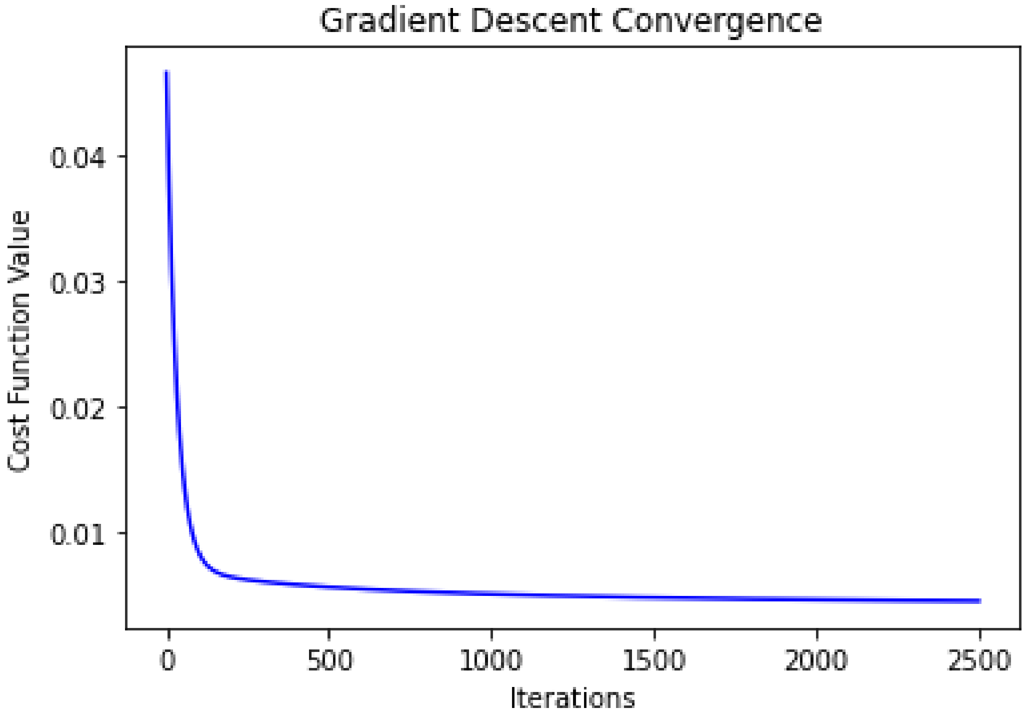
```
# create a NumPy array 'X_T' containing the transpose of the feature matrix 'X'
X_T = np.array(X.T)

# retrieve the number of rows 'm' and the number of columns 'n' from the␣
 ↪feature matrix 'X'
m, n = X.shape

# display the values of 'm' and 'n'
print("Number of training examples (m): ", m)
print("Number of features (n): ", n)

# set the number of iterations for gradient descent
iterations = 2500

# create a counter variable 'count' and a NumPy array 'j' to store the cost␣
 ↪function values for each iteration
count = 0
j = np.zeros(shape=(iterations, 1), dtype=float)

# display the shape of the 'j' array
print("Shape of 'j' array: ", j.shape)
```

```
Number of training examples (m):  436
Number of features (n):  12
Shape of 'j' array:  (2500, 1)
```

[22]:

```
# set the initial iteration count to zero
count = 0
```

```
# create a NumPy array 'j' to store the cost function values for each iteration
j = np.zeros(shape=(iterations, 1), dtype=float)

# perform gradient descent for the specified number of iterations
while count < iterations:

    # calculate the predicted values 'h' using the current parameters 'theta'
    h = X.dot(theta)

    # calculate the cost function value 'j' using the current parameters 'theta'
    j[count] = (1/(2*m)) * np.sum((h-Y)**2)

    # calculate the gradient of the cost function with respect to 'theta'
    grad = (1/m) * X_T.dot(h-Y)

    # update the parameters 'theta' using the learning rate 'alpha' and the gradient 'grad'
    alpha = 0.01
    theta = theta - alpha * grad

    # increment the iteration count
    count += 1

# plot the cost function values over the iterations
plt.plot(j,'b-')
plt.xlabel('Iterations')
plt.ylabel('Cost Function Value')
plt.title('Gradient Descent Convergence')
plt.show()
```



[23]:
```
# extract the test set features into NumPy arrays
N_X1_t = np.array(test.area)
N_X2_t = np.array(test.bedrooms)
N_X3_t = np.array(test.bathrooms)
N_X4_t = np.array(test.stories)
N_X5_t = np.array(test.parking)
N_X6_t = np.array(test.mainroad)
N_X7_t = np.array(test.guestroom)
N_X8_t = np.array(test.hotwaterheating)
N_X9_t = np.array(test.airconditioning)
N_X10_t = np.array(test.prefarea)
N_X11_t = np.array(test.basement)
N_X0_t = np.ones(109)
```

[24]:
```
# stack the test set features into a design matrix
N_X = np.vstack([N_X0_t,N_X1_t,N_X2_t,N_X3_t,N_X4_t,N_X5_t,␣
 ↪N_X6_t,N_X7_t,N_X8_t,N_X9_t,N_X10_t,N_X11_t])
N_X_T = N_X.T
N_X = np.array(N_X_T)
N_x = scaler.fit_transform(N_X)
N_X = N_x
N_X.shape
```

[24]:
```
(109, 12)
```

[25]:
```
# initialize the parameters for the test set
N_theta = np.array([0.,0.,0.,0.,0,0.,0.,0.,0.,0.,0,0.])
N_theta = N_theta.reshape(12,1)
N_theta
```

[25]:
```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[26]:
```
# initialize the target variable for the test set
N_Y_t = np.array(test.price)
N_Y = N_Y_t
N_Y = N_Y_t.reshape(109,1)
N_y = scaler.fit_transform(N_Y)
N_Y=N_y
N_Y.shape
```

[26]:
```
(109, 1)
```

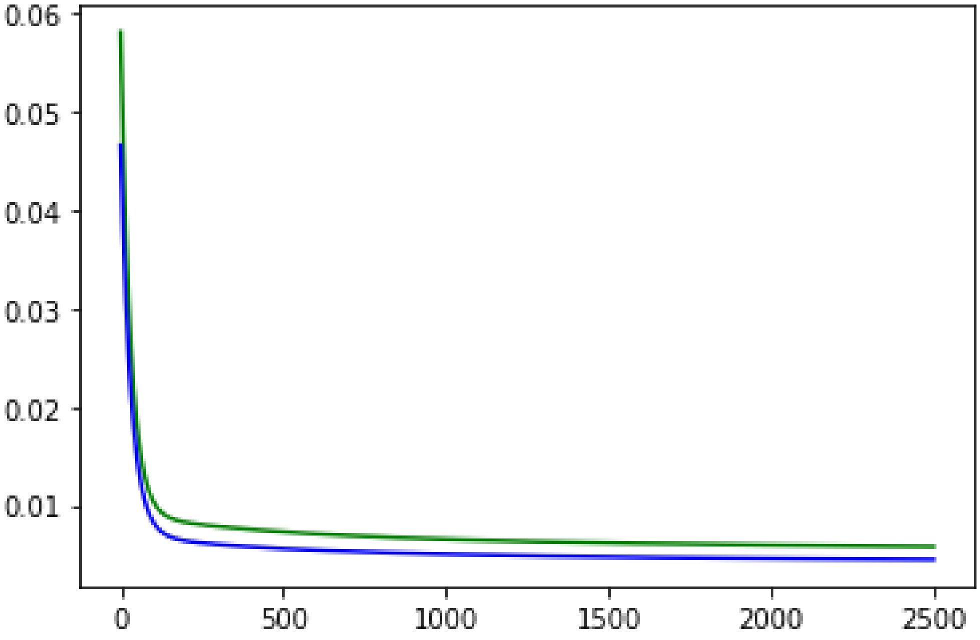
[27]:
```
N_X_T = np.array(N_X.T)
m,n = N_X.shape
m,n
```

[27]:
```
(109, 12)
```

[28]:
```
iterations = 2500
count=0
N_j = np.zeros(shape=(iterations, 1), dtype=float)

while(count < iterations):

    N_h_t = N_X.dot(N_theta)
    N_h = np.array(N_h_t)


    N_j[count]= (1/(2*m))*np.sum((N_h - N_Y)**2)
    grad_t = N_X_T.dot(N_h-N_Y)
    grad = grad_t*(1/m)

    N_theta = N_theta - 0.01*(grad)

    count += 1
```

[29]:
```
plt.plot(N_j,'g-')
plt.plot(j,'b-')
```

[29]:
```
[<matplotlib.lines.Line2D at 0x7fb76ca055b0>]
```



hw1-2b-standardization

February 20, 2023

```
[ ]: import numpy as np # import numpy library
     import pandas as pd # import pandas library
     import matplotlib.pyplot as plt # import matplotlib library
     from sklearn import preprocessing
```

```
[ ]: df = pd.read_csv("/content/sample_data/Housing.csv")

     # display DataFrame
     df
```

```
[ ]:         price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
     0    13300000  7420         4          2        3      yes        no       no
     1    12250000  8960         4          4        4      yes        no       no
     2    12250000  9960         3          2        2      yes        no      yes
     3    12215000  7500         4          2        2      yes        no      yes
     4    11410000  7420         4          1        2      yes       yes      yes
     ..        …     …         …          …        …        …         …        …
     540   1820000  3000         2          1        1      yes        no      yes
     541   1767150  2400         3          1        1       no        no       no
     542   1750000  3620         2          1        1      yes        no       no
     543   1750000  2910         3          1        1       no        no       no
     544   1750000  3850         3          1        2      yes        no       no

         hotwaterheating airconditioning  parking prefarea furnishingstatus
     0                no             yes        2      yes        furnished
     1                no             yes        3       no        furnished
     2                no              no        2      yes   semi-furnished
     3                no             yes        3      yes        furnished
     4                no             yes        2       no        furnished
     ..              …               …        …        …              …
     540              no              no        2       no      unfurnished
     541              no              no        0       no   semi-furnished
     542              no              no        0       no      unfurnished
     543              no              no        0       no        furnished
     544              no              no        0       no      unfurnished

     [545 rows x 13 columns]
```

[ ]:

```
df = df.replace(to_replace=['yes', 'no'], value=[1, 0])
df
```

[ ]:

```
        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0    13300000  7420         4          2        3         1          0
1    12250000  8960         4          4        4         1          0
2    12250000  9960         3          2        2         1          0
3    12215000  7500         4          2        2         1          0
4    11410000  7420         4          1        2         1          1
..        …   …       …        …      …       …        …
540   1820000  3000         2          1        1         1          0
541   1767150  2400         3          1        1         0          0
542   1750000  3620         2          1        1         1          0
543   1750000  2910         3          1        1         0          0
544   1750000  3850         3          1        2         1          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
0           0                0                1        2         1
1           0                0                1        3         0
2           1                0                0        2         1
3           1                0                1        3         1
4           1                0                1        2         0
..        …              …              …      …       …
540         1                0                0        2         0
541         0                0                0        0         0
542         0                0                0        0         0
543         0                0                0        0         0
544         0                0                0        0         0

    furnishingstatus
0          furnished
1          furnished
2     semi-furnished
3          furnished
4          furnished
..               …
540      unfurnished
541   semi-furnished
542      unfurnished
543        furnished
544      unfurnished

[545 rows x 13 columns]
```

[ ]:

```
# create a training set by randomly selecting 80% of the rows from the DataFrame
train = df.sample(frac=0.8, random_state=1)
```

```
# create a test set by dropping the rows in the training set from the DataFrame
test = df.drop(train.index)
```

[ ]:
```
train
```

[ ]:
```
       price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
62   7070000   6240         4          2        2         1          0
247  4550000   8400         4          1        4         1          0
142  5600000  10500         4          2        2         1          0
107  6125000   6420         3          1        3         1          0
483  2940000   6615         3          1        2         1          0
..       …      …         …          …        …         …
359  3710000   3600         3          1        1         1          0
36   8043000   7482         3          2        3         1          0
30   8400000   7475         3          2        4         1          0
20   8750000   4320         3          1        2         1          0
527  2275000   1836         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
62          0                0                1        1         0
247         0                0                0        3         0
142         0                0                0        1         0
107         1                0                0        0         1
483         0                0                0        0         0
..        …              …              …      …      …
359         0                0                0        1         0
36          0                1                0        1         1
30          0                0                1        2         0
20          1                1                0        2         0
527         1                0                0        0         0

    furnishingstatus
62         furnished
247      unfurnished
142   semi-furnished
107      unfurnished
483   semi-furnished
..              …
359      unfurnished
36         furnished
30       unfurnished
20    semi-furnished
527   semi-furnished

[436 rows x 13 columns]
```

[ ]:
```
test
```

[ ]:

```
        price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
2    12250000   9960         3          2        2         1          0
3    12215000   7500         4          2        2         1          0
7    10150000  16200         5          3        2         1          0
15    9100000   6000         4          1        2         1          0
22    8645000   8050         3          1        1         1          1
..        …      …         …          …        …         …          …
508   2590000   4400         2          1        1         1          0
513   2485000   4400         3          1        2         1          0
520   2450000   7700         2          1        1         1          0
537   1890000   1700         3          1        2         1          0
539   1855000   2990         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
2           1                0                0        2         1
3           1                0                1        3         1
7           0                0                0        0         0
15          1                0                0        2         0
22          1                0                1        1         0
..        …                …                …        …         …
508         0                0                0        0         0
513         0                0                0        0         0
520         0                0                0        0         0
537         0                0                0        0         0
539         0                0                0        1         0

    furnishingstatus
2     semi-furnished
3          furnished
7        unfurnished
15    semi-furnished
22         furnished
..               …
508      unfurnished
513      unfurnished
520      unfurnished
537      unfurnished
539      unfurnished

[109 rows x 13 columns]
```

[ ]:

```
# select specific columns for the training set
train = train[['price','area','bedrooms','bathrooms','stories','parking',␣
 ↪'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',␣
 ↪'prefarea']]
# select specific columns for the test set
```

```
test = test[['price','area','bedrooms','bathrooms','stories','parking',␣
 ↪'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',␣
 ↪'prefarea']]
```

[ ]:

```
scaler = preprocessing.StandardScaler()
```

[ ]:

```
train
```

[ ]:

```
       price   area  bedrooms  bathrooms  stories  parking  mainroad  \
62   7070000   6240         4          2        2        1         1
247  4550000   8400         4          1        4        3         1
142  5600000  10500         4          2        2        1         1
107  6125000   6420         3          1        3        0         1
483  2940000   6615         3          1        2        0         1
..       …      …         …          …        …        …         …
359  3710000   3600         3          1        1        1         1
36   8043000   7482         3          2        3        1         1
30   8400000   7475         3          2        4        2         1
20   8750000   4320         3          1        2        2         1
527  2275000   1836         2          1        1        0         0

     guestroom  basement  hotwaterheating  airconditioning  prefarea
62           0         0                0                1         0
247          0         0                0                0         0
142          0         0                0                0         0
107          0         1                0                0         1
483          0         0                0                0         0
..         …         …              …                …         …
359          0         0                0                0         0
36           0         0                1                0         1
30           0         0                0                1         0
20           0         1                1                0         0
527          0         1                0                0         0

[436 rows x 12 columns]
```

[ ]:

```
# create a NumPy array of the 'area' column from the training set
X1_t = np.array(train['area'])

# display the NumPy array
X1_t
```

[ ]:

```
array([ 6240,  8400, 10500,  6420,  6615,  3600,  3240,  6600,  2700,
        5000,  2650,  4775,  4800,  3700,  7700,  7420,  4280,  6000,
        6600,  3649,  3420,  5500,  3630,  3180,  3600,  8400,  3000,
        8880,  5750,  2145,  6360,  6525,  1950,  5850,  8372,  2870,
        4990,  2684,  5200,  6321,  4960,  3480,  3210,  4950,  6840,
        4350,  5850,  4410,  2500,  3850,  3180,  3162,  3500,  4340,
        6440,  5010,  3000,  4920,  3760,  3816,  6000,  7000,  3640,
        4080,  4160,  2910,  6060,  3000,  2787,  4815,  4785,  6600,
        5300,  3600,  6000,  2176,  3000,  7420,  7020,  3480,  5960,
        3510,  6420,  6450,  6210,  4500,  3000,  3180,  5700,  3520,
        4040,  5800,  2800,  6480,  4960,  4260,  7500,  5880, 10500,
        4500,  3850,  8500,  3120,  3990,  4095,  4800, 13200,  7770,
        6100,  4075,  6550,  4100,  4370,  3180,  7350,  3510,  3640,
        5500,  8250,  6600,  8250,  2475,  3850,  4500,  3720,  4360,
       10240,  5500,  3970,  3450,  3850,  5500,  3520,  2145,  6600,
        3640,  3986,  2953,  8250,  4130,  8580,  6000,  3500,  5885,
        7680,  2430,  3150,  6450,  8100,  5500,  1650,  3040,  4079,
        2747,  4600,  2325,  7231,  3520,  2145,  3450,  3620,  4000,
        6000,  6000,  4500,  3540,  7200,  3120,  4000,  2015,  4040,
        8000,  2787,  3512,  3420,  6060,  4500,  6360,  5450,  8250,
        3960,  7410, 10360,  3630,  6020,  4100,  6254,  4500,  4560,
        6710,  3500,  8880,  3600,  7152,  6000,  4040,  4000,  4040,
        5360,  6600,  3800,  3960,  4900,  3480,  3584,  2275,  4000,
        6500, 10500,  8960,  3290,  8875,  8580,  3450,  6600,  2800,
        5640,  3745, 10269,  6100, 12090,  5880,  6750,  6000,  5320,
        4000,  4040, 15600,  3090,  3970,  5450,  4770,  4095,  6000,
        6540,  6550,  4320,  3100,  4050,  3650,  3850,  5600,  2817,
        4510,  3000,  4995, 11410,  3000,  4840,  3600,  4000,  3500,
        7800,  5300,  4840,  3000,  3480,  2970,  5828,  3800,  4040,
       10700,  7320,  5000,  6325,  2880,  4300,  3150,  4000,  9500,
        4500,  3420,  3180,  2145,  5400,  3630,  6750,  4820,  5136,
        4120,  6825,  4600,  6650,  5800,  5720,  5000,  4352,  3300,
        2160,  5900,  3000,  4500,  3350,  5400,  4600,  9800,  3630,
        2610,  9667,  3635,  4000,  3180,  3630,  6600,  2610,  4960,
        5150,  6000,  3640,  2910,  3650,  3450,  4032,  7980,  1905,
        6000,  3360,  9620,  1950, 12900,  3240,  4320,  6540,  6000,
        7440,  3760,  8100,  4880,  6000,  2000,  5200,  4050,  9166,
        7950,  5500,  2700,  6000,  6900,  3500,  5076,  5985,  4300,
        8050,  5320,  5960,  7000,  7260,  6360,  3000,  3460, 12944,
        3880,  2400,  4080,  6000,  4500,  6050,  7000,  3930,  4600,
        7155,  4100,  2400,  3460,  4632,  4200,  4640,  8800,  3000,
        6300,  7000,  7000,  6900,  3420,  3264,  2640,  3150,  4320,
        6862, 11440,  4992,  3069,  3185,  3750,  5300,  7200,  6400,
        6800,  3400,  6420,  3792,  5500,  4600,  6800,  6000,  8520,
        6480,  8150,  5948,  3185,  5830,  3410,  3000,  8400,  6350,
        8100,  4800,  2856,  3185,  3780,  3640,  6000,  6000,  4800,
        5800,  6360,  4120,  5400,  2850,  5400,  2145,  4500,  3240,
       13200,  3900,  9000,  4646,  3840,  9000,  3520,  3640,  3600,
        7482,  7475,  4320,  1836])
```

[ ]:
```
X2_t = np.array(train.bedrooms)
X3_t = np.array(train.bathrooms)
```

```
X4_t = np.array(train.stories)
X5_t = np.array(train.parking)
X6_t = np.array(train.mainroad)
X7_t = np.array(train.guestroom)
X8_t = np.array(train.hotwaterheating)
X9_t = np.array(train.airconditioning)
X10_t = np.array(train.prefarea)
X11_t = np.array(train.basement)
X0_t= np.ones(436)
```

```
[ ]: X = np.vstack([X0_t,X1_t,X2_t,X3_t,X4_t,X5_t,X6_t,X7_t,X8_t,X9_t,X10_t,X11_t])
     X = X.T
     X = np.array(X)
     X
```

```
[ ]: array([[1.000e+00, 6.240e+03, 4.000e+00, …, 1.000e+00, 0.000e+00,
             0.000e+00],
            [1.000e+00, 8.400e+03, 4.000e+00, …, 0.000e+00, 0.000e+00,
             0.000e+00],
            [1.000e+00, 1.050e+04, 4.000e+00, …, 0.000e+00, 0.000e+00,
             0.000e+00],
            …,
            [1.000e+00, 7.475e+03, 3.000e+00, …, 1.000e+00, 0.000e+00,
             0.000e+00],
            [1.000e+00, 4.320e+03, 3.000e+00, …, 0.000e+00, 0.000e+00,
             1.000e+00],
            [1.000e+00, 1.836e+03, 2.000e+00, …, 0.000e+00, 0.000e+00,
             1.000e+00]])
```

```
[ ]: # scale the feature matrix using the fit_transform() method of the scaler object
     X_scaled = scaler.fit_transform(X)

     # assign the scaled feature matrix to the original variable name 'X'
     X = X_scaled
```

```
[ ]: # create a 1D NumPy array of zeros with length 12
     theta = np.zeros(12)

     # reshape the 1D array to a column vector using np.reshape
     theta = np.reshape(theta, (12,1))

     # display the column vector
     theta
```

```
[ ]: array([[0.],
            [0.],
            [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[ ]:

```
# create a 1D NumPy array 'Y_t' from the 'price' column of the training set␣
 ↪'train'
#h = np.matmul(X,theta)
Y_t = np.array(train.price)
Y = Y_t
Y = Y_t.reshape(436,1)
y = scaler.fit_transform(Y)
Y=y
```

[ ]:

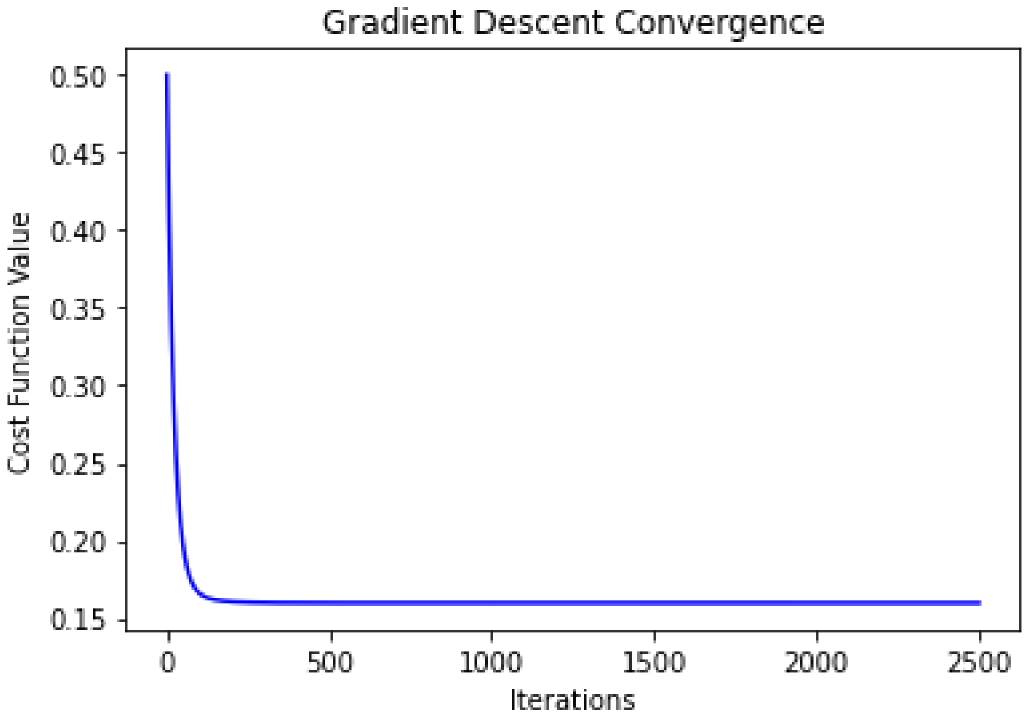
```
# create a NumPy array 'X_T' containing the transpose of the feature matrix 'X'
X_T = np.array(X.T)

# retrieve the number of rows 'm' and the number of columns 'n' from the␣
 ↪feature matrix 'X'
m, n = X.shape

# display the values of 'm' and 'n'
print("Number of training examples (m): ", m)
print("Number of features (n): ", n)

# set the number of iterations for gradient descent
iterations = 2500

# create a counter variable 'count' and a NumPy array 'j' to store the cost␣
 ↪function values for each iteration
count = 0
j = np.zeros(shape=(iterations, 1), dtype=float)

# display the shape of the 'j' array
print("Shape of 'j' array: ", j.shape)
```

```
Number of training examples (m):  436
Number of features (n):  12
Shape of 'j' array:  (2500, 1)
```

[ ]:

```
# set the initial iteration count to zero
count = 0
```

```
# create a NumPy array 'j' to store the cost function values for each iteration
j = np.zeros(shape=(iterations, 1), dtype=float)

# perform gradient descent for the specified number of iterations
while count < iterations:

    # calculate the predicted values 'h' using the current parameters 'theta'
    h = X.dot(theta)

    # calculate the cost function value 'j' using the current parameters 'theta'
    j[count] = (1/(2*m)) * np.sum((h-Y)**2)

    # calculate the gradient of the cost function with respect to 'theta'
    grad = (1/m) * X_T.dot(h-Y)

    # update the parameters 'theta' using the learning rate 'alpha' and the␣
 ↪gradient 'grad'
    alpha = 0.01
    theta = theta - alpha * grad

    # increment the iteration count
    count += 1

# plot the cost function values over the iterations
plt.plot(j,'b-')
plt.xlabel('Iterations')
plt.ylabel('Cost Function Value')
plt.title('Gradient Descent Convergence')
plt.show()
```



[ ]:

```
# extract the test set features into NumPy arrays
N_X1_t = np.array(test.area)
N_X2_t = np.array(test.bedrooms)
N_X3_t = np.array(test.bathrooms)
N_X4_t = np.array(test.stories)
N_X5_t = np.array(test.parking)
N_X6_t = np.array(test.mainroad)
N_X7_t = np.array(test.guestroom)
N_X8_t = np.array(test.hotwaterheating)
N_X9_t = np.array(test.airconditioning)
N_X10_t = np.array(test.prefarea)
N_X11_t = np.array(test.basement)
N_X0_t = np.ones(109)
```

[ ]:

```
# stack the test set features into a design matrix
N_X = np.vstack([N_X0_t,N_X1_t,N_X2_t,N_X3_t,N_X4_t,N_X5_t,␣
 ↪N_X6_t,N_X7_t,N_X8_t,N_X9_t,N_X10_t,N_X11_t])
N_X_T = N_X.T
N_X = np.array(N_X_T)
N_x = scaler.fit_transform(N_X)
N_X = N_x
N_X.shape
```

```
[ ]: (109, 12)
```

```
[ ]: # initialize the parameters for the test set
     N_theta = np.array([0.,0.,0.,0.,0,0.,0.,0.,0.,0.,0,0.])
     N_theta = N_theta.reshape(12,1)
     N_theta
```

```
[ ]: array([[0.],
            [0.],
            [0.],
            [0.],
            [0.],
            [0.],
            [0.],
            [0.],
            [0.],
            [0.],
            [0.],
            [0.]])
```

```
[ ]: # initialize the target variable for the test set
     N_Y_t = np.array(test.price)
     N_Y = N_Y_t
     N_Y = N_Y_t.reshape(109,1)
     N_y = scaler.fit_transform(N_Y)
     N_Y=N_y
     N_Y.shape
```

```
[ ]: (109, 1)
```

```
[ ]: N_X_T = np.array(N_X.T)
     m,n = N_X.shape
     m,n
```

```
[ ]: (109, 12)
```

```
[ ]: iterations = 2500
     count=0
     N_j = np.zeros(shape=(iterations, 1), dtype=float)

     while(count < iterations):

         N_h_t = N_X.dot(N_theta)
         N_h = np.array(N_h_t)


         N_j[count]= (1/(2*m))*np.sum((N_h - N_Y)**2)
```

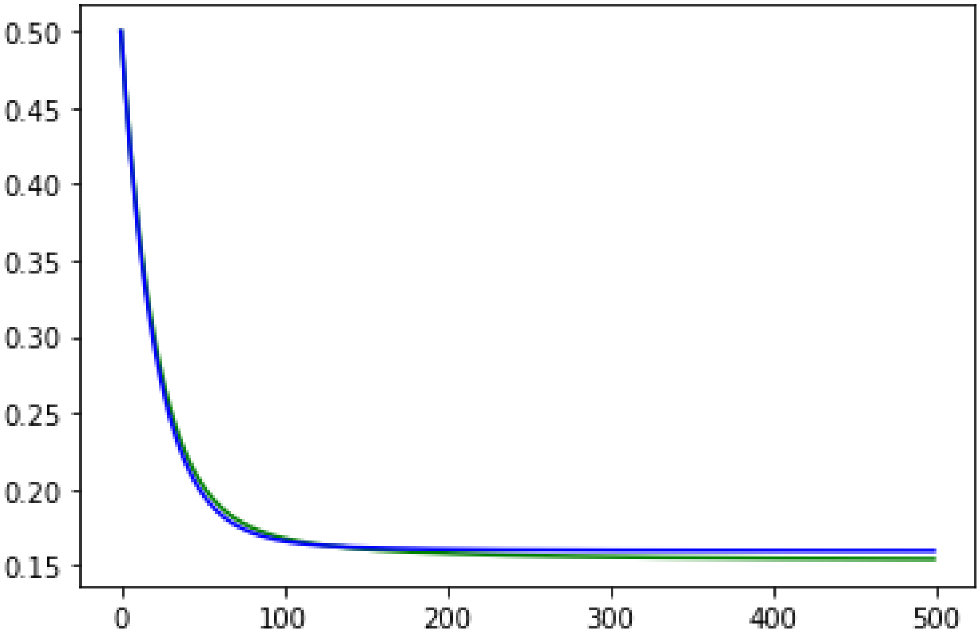
```
    grad_t = N_X_T.dot(N_h-N_Y)
    grad = grad_t*(1/m)

    N_theta = N_theta - 0.01*(grad)

    count += 1
```

[ ]:

```
plt.plot(N_j[:500],'g-')
plt.plot(j[:500],'b-')
```

[ ]:

```
[<matplotlib.lines.Line2D at 0x7f29d18e6df0>]
```



hw1-3a-normalization

February 20, 2023

[29]:
```
import numpy as np # import numpy library
import pandas as pd # import pandas library
import matplotlib.pyplot as plt # import matplotlib library
from sklearn import preprocessing  # import scikit-learn library (source: https:
 ↪//scikit-learn.org/stable/index.html)
```

[30]:
```
df = pd.read_csv("/content/sample_data/Housing.csv")

# display DataFrame
df
```

[30]:
```
        price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0    13300000  7420         4          2        3      yes        no       no
1    12250000  8960         4          4        4      yes        no       no
2    12250000  9960         3          2        2      yes        no      yes
3    12215000  7500         4          2        2      yes        no      yes
4    11410000  7420         4          1        2      yes       yes      yes
..        …     …       …          …        …        …         …
540   1820000  3000         2          1        1      yes        no      yes
541   1767150  2400         3          1        1       no        no       no
542   1750000  3620         2          1        1      yes        no       no
543   1750000  2910         3          1        1       no        no       no
544   1750000  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
0                no             yes        2      yes        furnished
1                no             yes        3       no        furnished
2                no              no        2      yes   semi-furnished
3                no             yes        3      yes        furnished
4                no             yes        2       no        furnished
..              …             …      …       …              …
540              no              no        2       no      unfurnished
541              no              no        0       no   semi-furnished
542              no              no        0       no      unfurnished
543              no              no        0       no        furnished
544              no              no        0       no      unfurnished
```

```
[545 rows x 13 columns]
```

[31]:
```
df = df.replace(to_replace=['yes', 'no'], value=[1, 0])
df
```

[31]:
```
        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0    13300000  7420         4          2        3         1          0
1    12250000  8960         4          4        4         1          0
2    12250000  9960         3          2        2         1          0
3    12215000  7500         4          2        2         1          0
4    11410000  7420         4          1        2         1          1
..        …   …       …        …      …       …        …
540   1820000  3000         2          1        1         1          0
541   1767150  2400         3          1        1         0          0
542   1750000  3620         2          1        1         1          0
543   1750000  2910         3          1        1         0          0
544   1750000  3850         3          1        2         1          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
0           0                0                1        2         1
1           0                0                1        3         0
2           1                0                0        2         1
3           1                0                1        3         1
4           1                0                1        2         0
..        …              …              …      …        …
540         1                0                0        2         0
541         0                0                0        0         0
542         0                0                0        0         0
543         0                0                0        0         0
544         0                0                0        0         0

    furnishingstatus
0          furnished
1          furnished
2     semi-furnished
3          furnished
4          furnished
..               …
540      unfurnished
541   semi-furnished
542      unfurnished
543        furnished
544      unfurnished

[545 rows x 13 columns]
```

[32]:
```
# create a training set by randomly selecting 80% of the rows from the DataFrame
train = df.sample(frac=0.8, random_state=1)

# create a test set by dropping the rows in the training set from the DataFrame
test = df.drop(train.index)
```

[33]:
```
test
```

[33]:
```
        price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
2    12250000   9960         3          2        2         1          0
3    12215000   7500         4          2        2         1          0
7    10150000  16200         5          3        2         1          0
15    9100000   6000         4          1        2         1          0
22    8645000   8050         3          1        1         1          1
..        …      …       …         …        …        …          …
508   2590000   4400         2          1        1         1          0
513   2485000   4400         3          1        2         1          0
520   2450000   7700         2          1        1         1          0
537   1890000   1700         3          1        2         1          0
539   1855000   2990         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
2           1                0                0        2         1
3           1                0                1        3         1
7           0                0                0        0         0
15          1                0                0        2         0
22          1                0                1        1         0
..        …              …                …      …        …
508         0                0                0        0         0
513         0                0                0        0         0
520         0                0                0        0         0
537         0                0                0        0         0
539         0                0                0        1         0

    furnishingstatus
2     semi-furnished
3          furnished
7        unfurnished
15    semi-furnished
22         furnished
..               …
508      unfurnished
513      unfurnished
520      unfurnished
537      unfurnished
539      unfurnished
```

```
[109 rows x 13 columns]
```

[34]:

```
# select specific columns for the training set
train = train[['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']]

# select specific columns for the test set
test = test[['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']]
```

[35]:

```
import sklearn.preprocessing # import scikit-learn library for data␣
 ↪preprocessing

# create an instance of the MinMaxScaler class for scaling features to a range␣
 ↪of [0, 1]
scaler = sklearn.preprocessing.MinMaxScaler()
```

[36]:

```
train
```

[36]:

```
       price   area  bedrooms  bathrooms  stories  parking
62   7070000   6240         4          2        2        1
247  4550000   8400         4          1        4        3
142  5600000  10500         4          2        2        1
107  6125000   6420         3          1        3        0
483  2940000   6615         3          1        2        0
..       …      …       …         …       …       …
359  3710000   3600         3          1        1        1
36   8043000   7482         3          2        3        1
30   8400000   7475         3          2        4        2
20   8750000   4320         3          1        2        2
527  2275000   1836         2          1        1        0

[436 rows x 6 columns]
```

[37]:

```
# create a NumPy array of the 'area' column from the training set
X1_t = np.array(train['area'])

# display the NumPy array
X1_t
```

[37]:

```
array([ 6240,  8400, 10500,  6420,  6615,  3600,  3240,  6600,  2700,
        5000,  2650,  4775,  4800,  3700,  7700,  7420,  4280,  6000,
        6600,  3649,  3420,  5500,  3630,  3180,  3600,  8400,  3000,
        8880,  5750,  2145,  6360,  6525,  1950,  5850,  8372,  2870,
        4990,  2684,  5200,  6321,  4960,  3480,  3210,  4950,  6840,
        4350,  5850,  4410,  2500,  3850,  3180,  3162,  3500,  4340,
        6440,  5010,  3000,  4920,  3760,  3816,  6000,  7000,  3640,
        4080,  4160,  2910,  6060,  3000,  2787,  4815,  4785,  6600,
        5300,  3600,  6000,  2176,  3000,  7420,  7020,  3480,  5960,
        3510,  6420,  6450,  6210,  4500,  3000,  3180,  5700,  3520,
        4040,  5800,  2800,  6480,  4960,  4260,  7500,  5880, 10500,
        4500,  3850,  8500,  3120,  3990,  4095,  4800, 13200,  7770,
        6100,  4075,  6550,  4100,  4370,  3180,  7350,  3510,  3640,
        5500,  8250,  6600,  8250,  2475,  3850,  4500,  3720,  4360,
       10240,  5500,  3970,  3450,  3850,  5500,  3520,  2145,  6600,
        3640,  3986,  2953,  8250,  4130,  8580,  6000,  3500,  5885,
        7680,  2430,  3150,  6450,  8100,  5500,  1650,  3040,  4079,
        2747,  4600,  2325,  7231,  3520,  2145,  3450,  3620,  4000,
        6000,  6000,  4500,  3540,  7200,  3120,  4000,  2015,  4040,
        8000,  2787,  3512,  3420,  6060,  4500,  6360,  5450,  8250,
        3960,  7410, 10360,  3630,  6020,  4100,  6254,  4500,  4560,
        6710,  3500,  8880,  3600,  7152,  6000,  4040,  4000,  4040,
        5360,  6600,  3800,  3960,  4900,  3480,  3584,  2275,  4000,
        6500, 10500,  8960,  3290,  8875,  8580,  3450,  6600,  2800,
        5640,  3745, 10269,  6100, 12090,  5880,  6750,  6000,  5320,
        4000,  4040, 15600,  3090,  3970,  5450,  4770,  4095,  6000,
        6540,  6550,  4320,  3100,  4050,  3650,  3850,  5600,  2817,
        4510,  3000,  4995, 11410,  3000,  4840,  3600,  4000,  3500,
        7800,  5300,  4840,  3000,  3480,  2970,  5828,  3800,  4040,
       10700,  7320,  5000,  6325,  2880,  4300,  3150,  4000,  9500,
        4500,  3420,  3180,  2145,  5400,  3630,  6750,  4820,  5136,
        4120,  6825,  4600,  6650,  5800,  5720,  5000,  4352,  3300,
        2160,  5900,  3000,  4500,  3350,  5400,  4600,  9800,  3630,
        2610,  9667,  3635,  4000,  3180,  3630,  6600,  2610,  4960,
        5150,  6000,  3640,  2910,  3650,  3450,  4032,  7980,  1905,
        6000,  3360,  9620,  1950, 12900,  3240,  4320,  6540,  6000,
        7440,  3760,  8100,  4880,  6000,  2000,  5200,  4050,  9166,
        7950,  5500,  2700,  6000,  6900,  3500,  5076,  5985,  4300,
        8050,  5320,  5960,  7000,  7260,  6360,  3000,  3460, 12944,
        3880,  2400,  4080,  6000,  4500,  6050,  7000,  3930,  4600,
        7155,  4100,  2400,  3460,  4632,  4200,  4640,  8800,  3000,
        6300,  7000,  7000,  6900,  3420,  3264,  2640,  3150,  4320,
        6862, 11440,  4992,  3069,  3185,  3750,  5300,  7200,  6400,
        6800,  3400,  6420,  3792,  5500,  4600,  6800,  6000,  8520,
        6480,  8150,  5948,  3185,  5830,  3410,  3000,  8400,  6350,
        8100,  4800,  2856,  3185,  3780,  3640,  6000,  6000,  4800,
        5800,  6360,  4120,  5400,  2850,  5400,  2145,  4500,  3240,
       13200,  3900,  9000,  4646,  3840,  9000,  3520,  3640,  3600,
        7482,  7475,  4320,  1836])
```

[38]:

```
X1_t = np.array(train['area']) # 'area' column
X2_t = np.array(train['bedrooms']) # 'bedrooms' column
X3_t = np.array(train['bathrooms']) # 'bathrooms' column
X4_t = np.array(train['stories']) # 'stories' column
X5_t = np.array(train['parking']) # 'parking' column
```

```
# create a NumPy array of ones to represent the bias term
X0_t = np.ones(len(train))
```

[39]:
```
# stack the selected feature arrays vertically using np.vstack
X = np.vstack([X0_t, X1_t, X2_t, X3_t, X4_t, X5_t])

# transpose the stacked array to make it a 6 x 436 matrix
X = X.T

# convert the stacked array to a NumPy array
X = np.array(X)

# display the NumPy array
X
```

[39]:
```
array([[1.000e+00, 6.240e+03, 4.000e+00, 2.000e+00, 2.000e+00, 1.000e+00],
       [1.000e+00, 8.400e+03, 4.000e+00, 1.000e+00, 4.000e+00, 3.000e+00],
       [1.000e+00, 1.050e+04, 4.000e+00, 2.000e+00, 2.000e+00, 1.000e+00],
       …,
       [1.000e+00, 7.475e+03, 3.000e+00, 2.000e+00, 4.000e+00, 2.000e+00],
       [1.000e+00, 4.320e+03, 3.000e+00, 1.000e+00, 2.000e+00, 2.000e+00],
       [1.000e+00, 1.836e+03, 2.000e+00, 1.000e+00, 1.000e+00, 0.000e+00]])
```

[40]:
```
# scale the feature matrix using the fit_transform() method of the scaler object
X_scaled = scaler.fit_transform(X)

# assign the scaled feature matrix to the original variable name 'X'
X = X_scaled
```

[41]:
```
# create a 1D NumPy array of zeros with length 6
theta = np.zeros(6)

# reshape the 1D array to a column vector using np.reshape
theta = np.reshape(theta, (6,1))

# display the column vector
theta
```

[41]:
```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[42]:
```
# create a 1D NumPy array 'Y_t' from the 'price' column of the training set␣
 ↪'train'
```

```
Y_t = np.array(train.price)

# create a copy of 'Y_t' to prevent changing the original data
Y = Y_t.copy()

# reshape 'Y' to a column vector using np.reshape
Y = np.reshape(Y, (436,1))

# create an instance of the MinMaxScaler class for scaling the target variable to a range of [0, 1]
scaler = sklearn.preprocessing.MinMaxScaler()

# scale the target variable using the fit_transform() method of the scaler object
Y_scaled = scaler.fit_transform(Y)

# assign the scaled target variable to the original variable name 'Y'
Y = Y_scaled
```

[43]:

```
# create a NumPy array 'X_T' containing the transpose of the feature matrix 'X'
X_T = np.array(X.T)

# retrieve the number of rows 'm' and the number of columns 'n' from the feature matrix 'X'
m, n = X.shape

# display the values of 'm' and 'n'
print("Number of training examples (m): ", m)
print("Number of features (n): ", n)

# set the number of iterations for gradient descent
iterations = 2500

# create a counter variable 'count' and a NumPy array 'j' to store the cost function values for each iteration
count = 0
j = np.zeros(shape=(iterations, 1), dtype=float)

# display the shape of the 'j' array
print("Shape of 'j' array: ", j.shape)
```

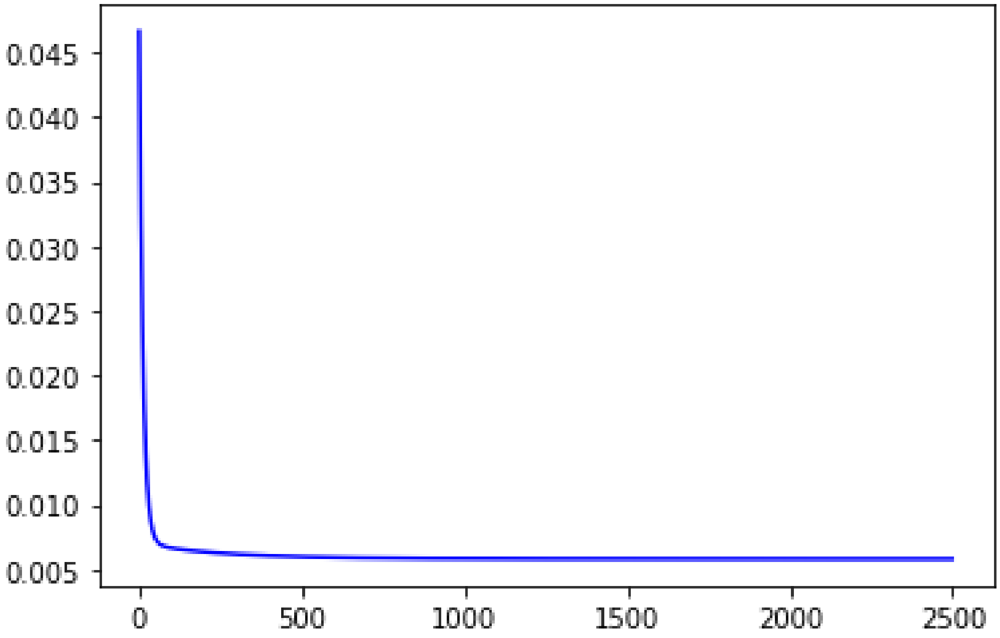
```
Number of training examples (m):  436
Number of features (n):  6
Shape of 'j' array:  (2500, 1)
```

[44]:

```
# Initialize variables
iterations = 2500
```

```python
count = 0
alpha = 0.1
lambda_ = 0.001
j = np.zeros(iterations)

# Perform gradient descent
while count < iterations:
    h_t = X.dot(theta)
    h = np.array(h_t, float)

    j[count] = (1/(2*m)) * np.sum((h - Y)**2) + (lambda_ / (2*m)) * np.
 ↪sum(theta**2)

    grad_t = X_T.dot(h - Y)
    grad = grad_t * (1/m)

    theta = theta * (1 - alpha * (lambda_ / m)) - alpha * grad

    count += 1

# Plot the cost function
plt.plot(j, 'b-')
```

```
[44]: [<matplotlib.lines.Line2D at 0x7f6e719f9700>]
```



```
[45]: # extract the test set features into NumPy arrays
      N_X1_t = np.array(test.area)
      N_X2_t = np.array(test.bedrooms)
      N_X3_t = np.array(test.bathrooms)
      N_X4_t = np.array(test.stories)
      N_X5_t = np.array(test.parking)
      N_X0_t = np.ones(109)
```

```
[46]: # stack the test set features into a design matrix
      N_X = np.vstack([N_X0_t,N_X1_t,N_X2_t,N_X3_t,N_X4_t,N_X5_t])
      N_X_T = N_X.T
      N_X = np.array(N_X_T)
      N_x = scaler.fit_transform(N_X)
      N_X = N_x
      N_X.shape
```

```
[46]: (109, 6)
```

```
[47]: N_theta = np.array([0.,0.,0.,0.,0,0.])
      N_theta = N_theta.reshape(6,1)
      N_theta
```

```
[47]: array([[0.],
             [0.],
             [0.],
             [0.],
             [0.],
             [0.]])
```

```
[48]: N_Y_t = np.array(test.price)
      N_Y = N_Y_t
      N_Y = N_Y_t.reshape(109,1)
      N_y = scaler.fit_transform(N_Y)
      N_Y=N_y
      N_Y.shape
```

```
[48]: (109, 1)
```

```
[49]: N_X_T = np.array(N_X.T)
      m,n = N_X.shape
      m,n
```

```
[49]: (109, 6)
```

```
[50]: iterations = 2500
      count=0
      N_j = np.zeros(shape=(iterations, 1), dtype=float)
```

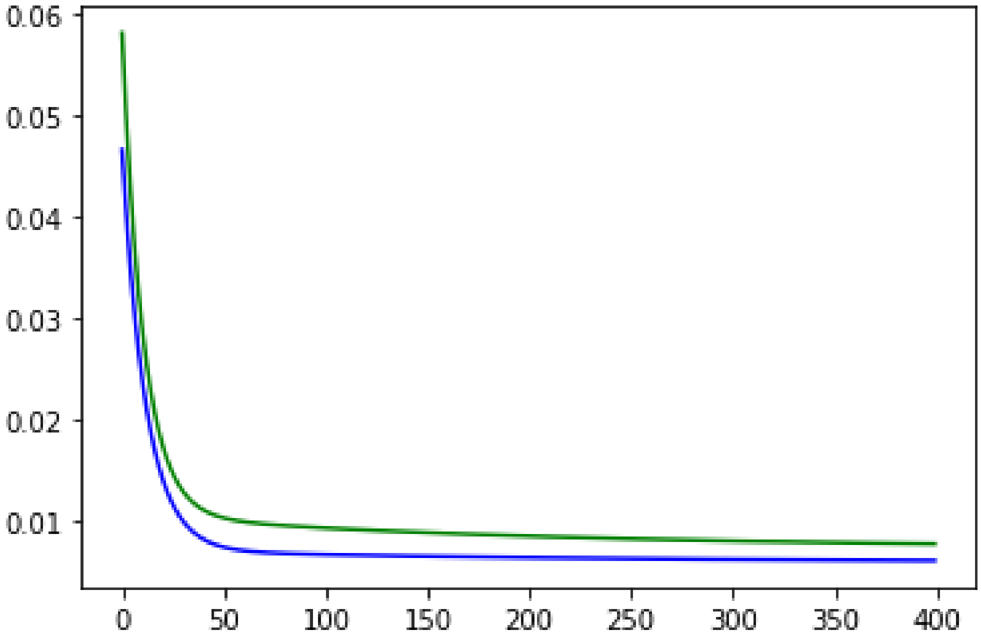
```
while(count < iterations):

    N_h_t = N_X.dot(N_theta)
    N_h = np.array(N_h_t)


    N_j[count]= (1/(2*m))*np.sum((N_h - N_Y)**2)+(0.001)*(1/(2*m))*np.
 ↪sum((N_theta)**2)

    grad_t = N_X_T.dot(N_h-N_Y)
    grad = grad_t*(1/m)

    N_theta = N_theta*(1-0.1*(0.001/m)) - 0.1*(grad)

    count += 1
```

[51]:

```
plt.plot(N_j[:400],'g-')
plt.plot(j[:400],'b-')
```

[51]:

```
[<matplotlib.lines.Line2D at 0x7f6e719beac0>]
```



hw1-3a-standardization

February 20, 2023

[1]:

```
import numpy as np # import numpy library
import pandas as pd # import pandas library
import matplotlib.pyplot as plt # import matplotlib library
from sklearn import preprocessing
```

[2]:

```
df = pd.read_csv("/content/sample_data/Housing.csv")

# display DataFrame
df
```

[2]:

```
        price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0    13300000  7420         4          2        3      yes        no       no
1    12250000  8960         4          4        4      yes        no       no
2    12250000  9960         3          2        2      yes        no      yes
3    12215000  7500         4          2        2      yes        no      yes
4    11410000  7420         4          1        2      yes       yes      yes
..        …     …       …         …       …       …        …        …
540   1820000  3000         2          1        1      yes        no      yes
541   1767150  2400         3          1        1       no        no       no
542   1750000  3620         2          1        1      yes        no       no
543   1750000  2910         3          1        1       no        no       no
544   1750000  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
0                no             yes        2      yes        furnished
1                no             yes        3       no        furnished
2                no              no        2      yes   semi-furnished
3                no             yes        3      yes        furnished
4                no             yes        2       no        furnished
..              …               …       …      …              …
540              no              no        2       no      unfurnished
541              no              no        0       no   semi-furnished
542              no              no        0       no      unfurnished
543              no              no        0       no        furnished
544              no              no        0       no      unfurnished

[545 rows x 13 columns]
```

[3]:

```
# create a training set by randomly selecting 80% of the rows from the DataFrame
train = df.sample(frac=0.8, random_state=1)

# create a test set by dropping the rows in the training set from the DataFrame
test = df.drop(train.index)
```

[4]:

```
train
```

[4]:

```
       price   area  bedrooms  bathrooms  stories mainroad guestroom basement  \
62   7070000   6240         4          2        2      yes        no       no
247  4550000   8400         4          1        4      yes        no       no
142  5600000  10500         4          2        2      yes        no       no
107  6125000   6420         3          1        3      yes        no      yes
483  2940000   6615         3          1        2      yes        no       no
..       …      …       …        …      …      …       …      …
359  3710000   3600         3          1        1      yes        no       no
36   8043000   7482         3          2        3      yes        no       no
30   8400000   7475         3          2        4      yes        no       no
20   8750000   4320         3          1        2      yes        no      yes
527  2275000   1836         2          1        1       no        no      yes

    hotwaterheating airconditioning  parking prefarea furnishingstatus
62               no             yes        1       no        furnished
247              no              no        3       no      unfurnished
142              no              no        1       no   semi-furnished
107              no              no        0      yes      unfurnished
483              no              no        0       no   semi-furnished
..              …             …      …      …              …
359              no              no        1       no      unfurnished
36              yes              no        1      yes        furnished
30               no             yes        2       no      unfurnished
20              yes              no        2       no   semi-furnished
527              no              no        0       no   semi-furnished

[436 rows x 13 columns]
```

[5]:

```
test
```

[5]:

```
        price   area  bedrooms  bathrooms  stories mainroad guestroom  \
2    12250000   9960         3          2        2      yes        no
3    12215000   7500         4          2        2      yes        no
7    10150000  16200         5          3        2      yes        no
15    9100000   6000         4          1        2      yes        no
22    8645000   8050         3          1        1      yes       yes
..        …      …       …        …      …      …       …
508   2590000   4400         2          1        1      yes        no
513   2485000   4400         3          1        2      yes        no
```

```
520   2450000   7700         2          1        1      yes        no
537   1890000   1700         3          1        2      yes        no
539   1855000   2990         2          1        1       no        no

    basement hotwaterheating airconditioning  parking prefarea   \
2        yes              no              no        2      yes
3        yes              no             yes        3      yes
7         no              no              no        0       no
15       yes              no              no        2       no
22       yes              no             yes        1       no
..       …             …             …      …      …
508       no              no              no        0       no
513       no              no              no        0       no
520       no              no              no        0       no
537       no              no              no        0       no
539       no              no              no        1       no

    furnishingstatus
2     semi-furnished
3          furnished
7        unfurnished
15    semi-furnished
22         furnished
..               …
508      unfurnished
513      unfurnished
520      unfurnished
537      unfurnished
539      unfurnished

[109 rows x 13 columns]
```

[6]:

```
# select specific columns for the training set
train = train[['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']]

# select specific columns for the test set
test = test[['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']]
```

[7]:

```
scaler = preprocessing.StandardScaler()
```

[8]:

```
# create a NumPy array of the 'area' column from the training set
X1_t = np.array(train['area'])

# display the NumPy array
X1_t
```

```
[8]: array([ 6240,  8400, 10500,  6420,  6615,  3600,  3240,  6600,  2700,
             5000,  2650,  4775,  4800,  3700,  7700,  7420,  4280,  6000,
             6600,  3649,  3420,  5500,  3630,  3180,  3600,  8400,  3000,
             8880,  5750,  2145,  6360,  6525,  1950,  5850,  8372,  2870,
             4990,  2684,  5200,  6321,  4960,  3480,  3210,  4950,  6840,
             4350,  5850,  4410,  2500,  3850,  3180,  3162,  3500,  4340,
             6440,  5010,  3000,  4920,  3760,  3816,  6000,  7000,  3640,
             4080,  4160,  2910,  6060,  3000,  2787,  4815,  4785,  6600,
             5300,  3600,  6000,  2176,  3000,  7420,  7020,  3480,  5960,
             3510,  6420,  6450,  6210,  4500,  3000,  3180,  5700,  3520,
             4040,  5800,  2800,  6480,  4960,  4260,  7500,  5880, 10500,
             4500,  3850,  8500,  3120,  3990,  4095,  4800, 13200,  7770,
             6100,  4075,  6550,  4100,  4370,  3180,  7350,  3510,  3640,
             5500,  8250,  6600,  8250,  2475,  3850,  4500,  3720,  4360,
            10240,  5500,  3970,  3450,  3850,  5500,  3520,  2145,  6600,
             3640,  3986,  2953,  8250,  4130,  8580,  6000,  3500,  5885,
             7680,  2430,  3150,  6450,  8100,  5500,  1650,  3040,  4079,
             2747,  4600,  2325,  7231,  3520,  2145,  3450,  3620,  4000,
             6000,  6000,  4500,  3540,  7200,  3120,  4000,  2015,  4040,
             8000,  2787,  3512,  3420,  6060,  4500,  6360,  5450,  8250,
             3960,  7410, 10360,  3630,  6020,  4100,  6254,  4500,  4560,
             6710,  3500,  8880,  3600,  7152,  6000,  4040,  4000,  4040,
             5360,  6600,  3800,  3960,  4900,  3480,  3584,  2275,  4000,
             6500, 10500,  8960,  3290,  8875,  8580,  3450,  6600,  2800,
             5640,  3745, 10269,  6100, 12090,  5880,  6750,  6000,  5320,
             4000,  4040, 15600,  3090,  3970,  5450,  4770,  4095,  6000,
             6540,  6550,  4320,  3100,  4050,  3650,  3850,  5600,  2817,
             4510,  3000,  4995, 11410,  3000,  4840,  3600,  4000,  3500,
             7800,  5300,  4840,  3000,  3480,  2970,  5828,  3800,  4040,
            10700,  7320,  5000,  6325,  2880,  4300,  3150,  4000,  9500,
             4500,  3420,  3180,  2145,  5400,  3630,  6750,  4820,  5136,
             4120,  6825,  4600,  6650,  5800,  5720,  5000,  4352,  3300,
             2160,  5900,  3000,  4500,  3350,  5400,  4600,  9800,  3630,
             2610,  9667,  3635,  4000,  3180,  3630,  6600,  2610,  4960,
             5150,  6000,  3640,  2910,  3650,  3450,  4032,  7980,  1905,
             6000,  3360,  9620,  1950, 12900,  3240,  4320,  6540,  6000,
             7440,  3760,  8100,  4880,  6000,  2000,  5200,  4050,  9166,
             7950,  5500,  2700,  6000,  6900,  3500,  5076,  5985,  4300,
             8050,  5320,  5960,  7000,  7260,  6360,  3000,  3460, 12944,
             3880,  2400,  4080,  6000,  4500,  6050,  7000,  3930,  4600,
             7155,  4100,  2400,  3460,  4632,  4200,  4640,  8800,  3000,
             6300,  7000,  7000,  6900,  3420,  3264,  2640,  3150,  4320,
             6862, 11440,  4992,  3069,  3185,  3750,  5300,  7200,  6400,
             6800,  3400,  6420,  3792,  5500,  4600,  6800,  6000,  8520,
             6480,  8150,  5948,  3185,  5830,  3410,  3000,  8400,  6350,
             8100,  4800,  2856,  3185,  3780,  3640,  6000,  6000,  4800,
             5800,  6360,  4120,  5400,  2850,  5400,  2145,  4500,  3240,
            13200,  3900,  9000,  4646,  3840,  9000,  3520,  3640,  3600,
             7482,  7475,  4320,  1836])
```

[9]:

```
X1_t = np.array(train['area']) # 'area' column
X2_t = np.array(train['bedrooms']) # 'bedrooms' column
X3_t = np.array(train['bathrooms']) # 'bathrooms' column
X4_t = np.array(train['stories']) # 'stories' column
X5_t = np.array(train['parking']) # 'parking' column

# create a NumPy array of ones to represent the bias term
X0_t = np.ones(len(train))
```

[10]:

```
# stack the selected feature arrays vertically using np.vstack
X = np.vstack([X0_t, X1_t, X2_t, X3_t, X4_t, X5_t])

# transpose the stacked array to make it a 6 x 436 matrix
X = X.T

# convert the stacked array to a NumPy array
X = np.array(X)

# display the NumPy array
X
```

[10]:

```
array([[1.000e+00, 6.240e+03, 4.000e+00, 2.000e+00, 2.000e+00, 1.000e+00],
       [1.000e+00, 8.400e+03, 4.000e+00, 1.000e+00, 4.000e+00, 3.000e+00],
       [1.000e+00, 1.050e+04, 4.000e+00, 2.000e+00, 2.000e+00, 1.000e+00],
       …,
       [1.000e+00, 7.475e+03, 3.000e+00, 2.000e+00, 4.000e+00, 2.000e+00],
       [1.000e+00, 4.320e+03, 3.000e+00, 1.000e+00, 2.000e+00, 2.000e+00],
       [1.000e+00, 1.836e+03, 2.000e+00, 1.000e+00, 1.000e+00, 0.000e+00]])
```

[11]:

```
x = scaler.fit_transform(X)
X = x
```

[12]:

```
# create a 1D NumPy array of zeros with length 6
theta = np.zeros(6)

# reshape the 1D array to a column vector using np.reshape
theta = np.reshape(theta, (6,1))

# display the column vector
theta
```

[12]:

```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[13]:

```
# create a 1D NumPy array 'Y_t' from the 'price' column of the training set␣
 ↪'train'
#h = np.matmul(X,theta)
Y_t = np.array(train.price)
Y = Y_t
Y = Y_t.reshape(436,1)
y = scaler.fit_transform(Y)
Y=y
```

[14]:

```
# create a NumPy array 'X_T' containing the transpose of the feature matrix 'X'
X_T = np.array(X.T)

# retrieve the number of rows 'm' and the number of columns 'n' from the␣
 ↪feature matrix 'X'
m, n = X.shape

# display the values of 'm' and 'n'
print("Number of training examples (m): ", m)
print("Number of features (n): ", n)

# set the number of iterations for gradient descent
iterations = 1000

# create a counter variable 'count' and a NumPy array 'j' to store the cost␣
 ↪function values for each iteration
count = 0
j = np.zeros(shape=(iterations, 1), dtype=float)

# display the shape of the 'j' array
print("Shape of 'j' array: ", j.shape)
```

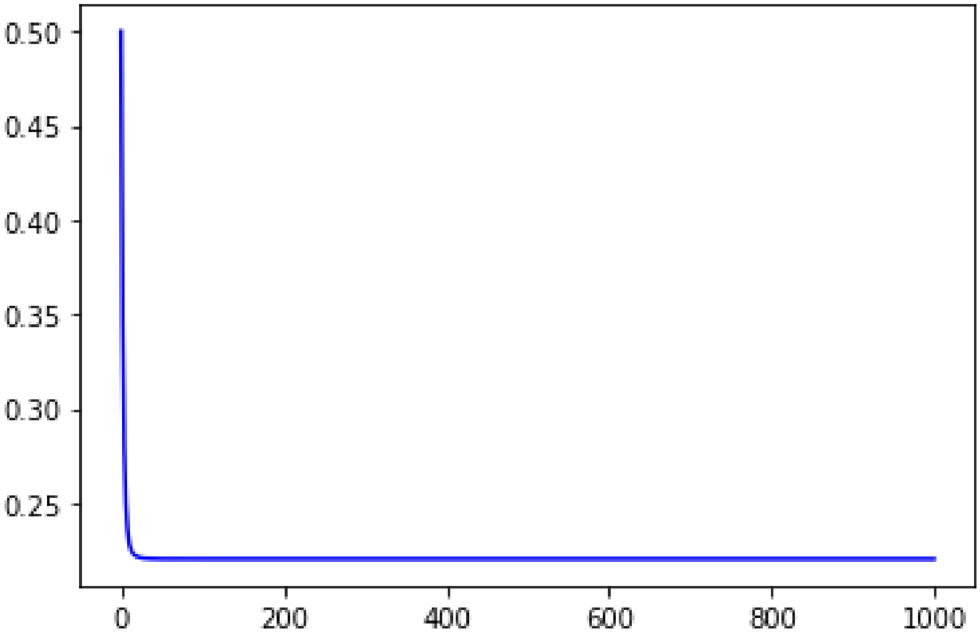
```
Number of training examples (m):  436
Number of features (n):  6
Shape of 'j' array:  (1000, 1)
```

[15]:

```
# Initialize variables
iterations = 1000
count = 0
alpha = 0.1
lambda_ = 0.001
j = np.zeros(iterations)

# Perform gradient descent
```

```
while count < iterations:
    h_t = X.dot(theta)
    h = np.array(h_t, float)

    j[count] = (1/(2*m)) * np.sum((h - Y)**2) + (lambda_ / (2*m)) * np.
 ↪sum(theta**2)

    grad_t = X_T.dot(h - Y)
    grad = grad_t * (1/m)

    theta = theta * (1 - alpha * (lambda_ / m)) - alpha * grad

    count += 1

# Plot the cost function
plt.plot(j, 'b-')
```

```
[15]: [<matplotlib.lines.Line2D at 0x7f153537e520>]
```



```
[16]: # extract the test set features into NumPy arrays
N_X1_t = np.array(test.area)
N_X2_t = np.array(test.bedrooms)
N_X3_t = np.array(test.bathrooms)
N_X4_t = np.array(test.stories)
N_X5_t = np.array(test.parking)
```

```
N_X0_t = np.ones(109)
```

[17]:
```
# stack the test set features into a design matrix
N_X = np.vstack([N_X0_t,N_X1_t,N_X2_t,N_X3_t,N_X4_t,N_X5_t])
N_X_T = N_X.T
N_X = np.array(N_X_T)
N_x = scaler.fit_transform(N_X)
N_X = N_x
N_X.shape
```

[17]:
```
(109, 6)
```

[18]:
```
N_theta = np.array([0.,0.,0.,0.,0,0.])
N_theta = N_theta.reshape(6,1)
N_theta
```

[18]:
```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[19]:
```
N_Y_t = np.array(test.price)
N_Y = N_Y_t
N_Y = N_Y_t.reshape(109,1)
N_y = scaler.fit_transform(N_Y)
N_Y=N_y
N_Y.shape
```

[19]:
```
(109, 1)
```

[20]:
```
N_X_T = np.array(N_X.T)
m,n = N_X.shape
m,n
```

[20]:
```
(109, 6)
```

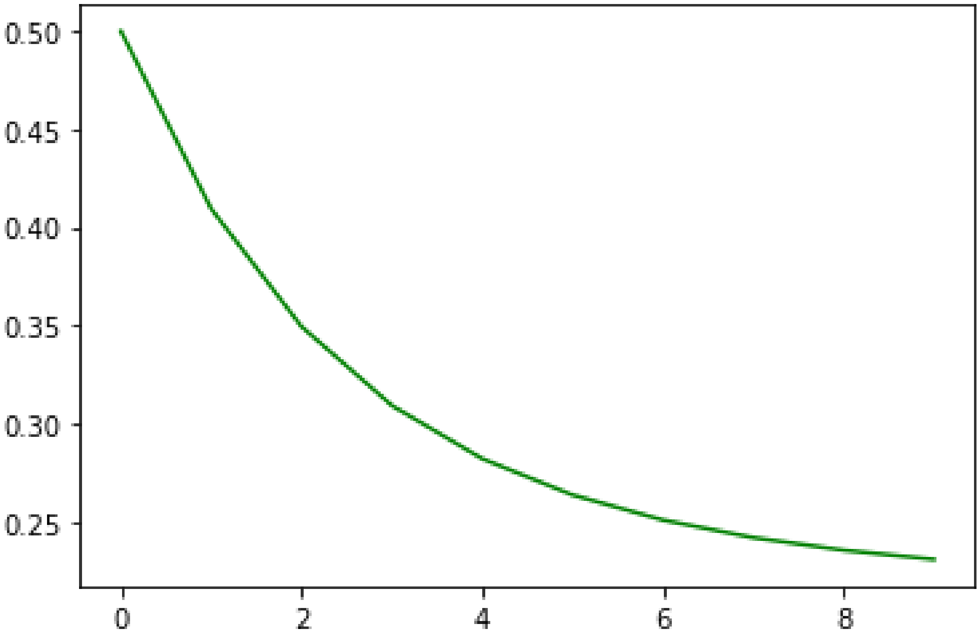
[22]:
```
iterations = 2500
count=0
N_j = np.zeros(shape=(iterations, 1), dtype=float)

while(count < iterations):

    N_h_t = N_X.dot(N_theta)
    N_h = np.array(N_h_t)
    N_j[count]= (1/(2*m))*np.sum((N_h - N_Y)**2)+(0.001)*(1/(2*m))*np.
 ↪sum((N_theta)**2)

    grad_t = N_X_T.dot(N_h-N_Y)
    grad = grad_t*(1/m)

    N_theta = N_theta*(1-0.1*(0.001/m)) - 0.1*(grad)

    count += 1
```

[25]:

```
plt.plot(N_j[:10],'g-')
#plt.plot(j[:10],'b-')
```

[25]:

```
[<matplotlib.lines.Line2D at 0x7f1534d59070>]
```



hw1-3b-normalization

February 20, 2023

```
[100]: import numpy as np # import numpy library
       import pandas as pd # import pandas library
       import matplotlib.pyplot as plt # import matplotlib library
       from sklearn import preprocessing  # import scikit-learn library (source: https:
        ↪//scikit-learn.org/stable/index.html)
```

```
[101]: df = pd.read_csv("/content/sample_data/Housing.csv")

       # display DataFrame
       df
```

```
[101]:        price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
       0    13300000  7420         4          2        3      yes        no       no
       1    12250000  8960         4          4        4      yes        no       no
       2    12250000  9960         3          2        2      yes        no      yes
       3    12215000  7500         4          2        2      yes        no      yes
       4    11410000  7420         4          1        2      yes       yes      yes
       ..        …     …         …          …        …        …        …
       540   1820000  3000         2          1        1      yes        no      yes
       541   1767150  2400         3          1        1       no        no       no
       542   1750000  3620         2          1        1      yes        no       no
       543   1750000  2910         3          1        1       no        no       no
       544   1750000  3850         3          1        2      yes        no       no

           hotwaterheating airconditioning  parking prefarea furnishingstatus
       0                no             yes        2      yes        furnished
       1                no             yes        3       no        furnished
       2                no              no        2      yes   semi-furnished
       3                no             yes        3      yes        furnished
       4                no             yes        2       no        furnished
       ..              …              …       …       …              …
       540              no              no        2       no      unfurnished
       541              no              no        0       no   semi-furnished
       542              no              no        0       no      unfurnished
       543              no              no        0       no        furnished
       544              no              no        0       no      unfurnished
```

```
[545 rows x 13 columns]
```

[102]:

```
df = df.replace(to_replace=['yes', 'no'], value=[1, 0])
df
```

[102]:

```
        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0    13300000  7420         4          2        3         1          0
1    12250000  8960         4          4        4         1          0
2    12250000  9960         3          2        2         1          0
3    12215000  7500         4          2        2         1          0
4    11410000  7420         4          1        2         1          1
..        …   …       …        …      …        …        …
540   1820000  3000         2          1        1         1          0
541   1767150  2400         3          1        1         0          0
542   1750000  3620         2          1        1         1          0
543   1750000  2910         3          1        1         0          0
544   1750000  3850         3          1        2         1          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
0           0                0                1        2         1
1           0                0                1        3         0
2           1                0                0        2         1
3           1                0                1        3         1
4           1                0                1        2         0
..        …              …              …      …        …
540         1                0                0        2         0
541         0                0                0        0         0
542         0                0                0        0         0
543         0                0                0        0         0
544         0                0                0        0         0

    furnishingstatus
0          furnished
1          furnished
2     semi-furnished
3          furnished
4          furnished
..                …
540      unfurnished
541   semi-furnished
542      unfurnished
543        furnished
544      unfurnished

[545 rows x 13 columns]
```

[103]:

```
# create a training set by randomly selecting 80% of the rows from the DataFrame
train = df.sample(frac=0.8, random_state=1)

# create a test set by dropping the rows in the training set from the DataFrame
test = df.drop(train.index)
```

[104]:

```
train
```

[104]:

```
       price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
62   7070000   6240         4          2        2         1          0
247  4550000   8400         4          1        4         1          0
142  5600000  10500         4          2        2         1          0
107  6125000   6420         3          1        3         1          0
483  2940000   6615         3          1        2         1          0
..       …      …       …        …      …       …        …
359  3710000   3600         3          1        1         1          0
36   8043000   7482         3          2        3         1          0
30   8400000   7475         3          2        4         1          0
20   8750000   4320         3          1        2         1          0
527  2275000   1836         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
62          0                0                1        1         0
247         0                0                0        3         0
142         0                0                0        1         0
107         1                0                0        0         1
483         0                0                0        0         0
..        …              …              …      …       …
359         0                0                0        1         0
36          0                1                0        1         1
30          0                0                1        2         0
20          1                1                0        2         0
527         1                0                0        0         0

    furnishingstatus
62         furnished
247      unfurnished
142   semi-furnished
107      unfurnished
483   semi-furnished
..               …
359      unfurnished
36         furnished
30       unfurnished
20    semi-furnished
527   semi-furnished
```

```
[436 rows x 13 columns]
```

[105]:
```
test
```

[105]:
```
        price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
2    12250000   9960         3          2        2         1          0
3    12215000   7500         4          2        2         1          0
7    10150000  16200         5          3        2         1          0
15    9100000   6000         4          1        2         1          0
22    8645000   8050         3          1        1         1          1
..        …      …       …         …       …        …         …
508   2590000   4400         2          1        1         1          0
513   2485000   4400         3          1        2         1          0
520   2450000   7700         2          1        1         1          0
537   1890000   1700         3          1        2         1          0
539   1855000   2990         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
2           1                0                0        2         1
3           1                0                1        3         1
7           0                0                0        0         0
15          1                0                0        2         0
22          1                0                1        1         0
..        …              …              …      …       …
508         0                0                0        0         0
513         0                0                0        0         0
520         0                0                0        0         0
537         0                0                0        0         0
539         0                0                0        1         0

    furnishingstatus
2     semi-furnished
3          furnished
7        unfurnished
15    semi-furnished
22         furnished
..               …
508      unfurnished
513      unfurnished
520      unfurnished
537      unfurnished
539      unfurnished

[109 rows x 13 columns]
```

[106]:
```
# select specific columns for the training set
```

```
train = train[['price','area','bedrooms','bathrooms','stories','parking',␣
 ↪'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',␣
 ↪'prefarea']]
# select specific columns for the test set
test = test[['price','area','bedrooms','bathrooms','stories','parking',␣
 ↪'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',␣
 ↪'prefarea']]
```

[107]:

```
import sklearn.preprocessing # import scikit-learn library for data␣
 ↪preprocessing

# create an instance of the MinMaxScaler class for scaling features to a range␣
 ↪of [0, 1]
scaler = sklearn.preprocessing.MinMaxScaler()
```

[108]:

```
train
```

[108]:

```
       price   area  bedrooms  bathrooms  stories  parking  mainroad  \
62   7070000   6240         4          2        2        1         1
247  4550000   8400         4          1        4        3         1
142  5600000  10500         4          2        2        1         1
107  6125000   6420         3          1        3        0         1
483  2940000   6615         3          1        2        0         1
..       …      …       …        …       …       …       …
359  3710000   3600         3          1        1        1         1
36   8043000   7482         3          2        3        1         1
30   8400000   7475         3          2        4        2         1
20   8750000   4320         3          1        2        2         1
527  2275000   1836         2          1        1        0         0

     guestroom  basement  hotwaterheating  airconditioning  prefarea
62           0         0                0                1         0
247          0         0                0                0         0
142          0         0                0                0         0
107          0         1                0                0         1
483          0         0                0                0         0
..         …       …              …              …       …
359          0         0                0                0         0
36           0         0                1                0         1
30           0         0                0                1         0
20           0         1                1                0         0
527          0         1                0                0         0

[436 rows x 12 columns]
```

[109]:

```
# create a NumPy array of the 'area' column from the training set
X1_t = np.array(train['area'])

# display the NumPy array
X1_t
```

[109]:

```
array([ 6240,  8400, 10500,  6420,  6615,  3600,  3240,  6600,  2700,
        5000,  2650,  4775,  4800,  3700,  7700,  7420,  4280,  6000,
        6600,  3649,  3420,  5500,  3630,  3180,  3600,  8400,  3000,
        8880,  5750,  2145,  6360,  6525,  1950,  5850,  8372,  2870,
        4990,  2684,  5200,  6321,  4960,  3480,  3210,  4950,  6840,
        4350,  5850,  4410,  2500,  3850,  3180,  3162,  3500,  4340,
        6440,  5010,  3000,  4920,  3760,  3816,  6000,  7000,  3640,
        4080,  4160,  2910,  6060,  3000,  2787,  4815,  4785,  6600,
        5300,  3600,  6000,  2176,  3000,  7420,  7020,  3480,  5960,
        3510,  6420,  6450,  6210,  4500,  3000,  3180,  5700,  3520,
        4040,  5800,  2800,  6480,  4960,  4260,  7500,  5880, 10500,
        4500,  3850,  8500,  3120,  3990,  4095,  4800, 13200,  7770,
        6100,  4075,  6550,  4100,  4370,  3180,  7350,  3510,  3640,
        5500,  8250,  6600,  8250,  2475,  3850,  4500,  3720,  4360,
       10240,  5500,  3970,  3450,  3850,  5500,  3520,  2145,  6600,
        3640,  3986,  2953,  8250,  4130,  8580,  6000,  3500,  5885,
        7680,  2430,  3150,  6450,  8100,  5500,  1650,  3040,  4079,
        2747,  4600,  2325,  7231,  3520,  2145,  3450,  3620,  4000,
        6000,  6000,  4500,  3540,  7200,  3120,  4000,  2015,  4040,
        8000,  2787,  3512,  3420,  6060,  4500,  6360,  5450,  8250,
        3960,  7410, 10360,  3630,  6020,  4100,  6254,  4500,  4560,
        6710,  3500,  8880,  3600,  7152,  6000,  4040,  4000,  4040,
        5360,  6600,  3800,  3960,  4900,  3480,  3584,  2275,  4000,
        6500, 10500,  8960,  3290,  8875,  8580,  3450,  6600,  2800,
        5640,  3745, 10269,  6100, 12090,  5880,  6750,  6000,  5320,
        4000,  4040, 15600,  3090,  3970,  5450,  4770,  4095,  6000,
        6540,  6550,  4320,  3100,  4050,  3650,  3850,  5600,  2817,
        4510,  3000,  4995, 11410,  3000,  4840,  3600,  4000,  3500,
        7800,  5300,  4840,  3000,  3480,  2970,  5828,  3800,  4040,
       10700,  7320,  5000,  6325,  2880,  4300,  3150,  4000,  9500,
        4500,  3420,  3180,  2145,  5400,  3630,  6750,  4820,  5136,
        4120,  6825,  4600,  6650,  5800,  5720,  5000,  4352,  3300,
        2160,  5900,  3000,  4500,  3350,  5400,  4600,  9800,  3630,
        2610,  9667,  3635,  4000,  3180,  3630,  6600,  2610,  4960,
        5150,  6000,  3640,  2910,  3650,  3450,  4032,  7980,  1905,
        6000,  3360,  9620,  1950, 12900,  3240,  4320,  6540,  6000,
        7440,  3760,  8100,  4880,  6000,  2000,  5200,  4050,  9166,
        7950,  5500,  2700,  6000,  6900,  3500,  5076,  5985,  4300,
        8050,  5320,  5960,  7000,  7260,  6360,  3000,  3460, 12944,
        3880,  2400,  4080,  6000,  4500,  6050,  7000,  3930,  4600,
        7155,  4100,  2400,  3460,  4632,  4200,  4640,  8800,  3000,
        6300,  7000,  7000,  6900,  3420,  3264,  2640,  3150,  4320,
        6862, 11440,  4992,  3069,  3185,  3750,  5300,  7200,  6400,
        6800,  3400,  6420,  3792,  5500,  4600,  6800,  6000,  8520,
        6480,  8150,  5948,  3185,  5830,  3410,  3000,  8400,  6350,
        8100,  4800,  2856,  3185,  3780,  3640,  6000,  6000,  4800,
        5800,  6360,  4120,  5400,  2850,  5400,  2145,  4500,  3240,
       13200,  3900,  9000,  4646,  3840,  9000,  3520,  3640,  3600,
        7482,  7475,  4320,  1836])
```

[110]:

```
X2_t = np.array(train.bedrooms)
X3_t = np.array(train.bathrooms)
X4_t = np.array(train.stories)
X5_t = np.array(train.parking)
X6_t = np.array(train.mainroad)
X7_t = np.array(train.guestroom)
X8_t = np.array(train.hotwaterheating)
X9_t = np.array(train.airconditioning)
X10_t = np.array(train.prefarea)
X11_t = np.array(train.basement)
X0_t= np.ones(436)
```

[111]:

```
# stack the selected feature arrays vertically using np.vstack
X = np.vstack([X0_t, X1_t, X2_t, X3_t, X4_t, X5_t,␣
 ↪X6_t,X7_t,X8_t,X9_t,X10_t,X11_t])

# transpose the stacked array to make it a 6 x 436 matrix
X = X.T

# convert the stacked array to a NumPy array
X = np.array(X)

# display the NumPy array
X
```

[111]:

```
array([[1.000e+00, 6.240e+03, 4.000e+00, …, 1.000e+00, 0.000e+00,
        0.000e+00],
       [1.000e+00, 8.400e+03, 4.000e+00, …, 0.000e+00, 0.000e+00,
        0.000e+00],
       [1.000e+00, 1.050e+04, 4.000e+00, …, 0.000e+00, 0.000e+00,
        0.000e+00],
       …,
       [1.000e+00, 7.475e+03, 3.000e+00, …, 1.000e+00, 0.000e+00,
        0.000e+00],
       [1.000e+00, 4.320e+03, 3.000e+00, …, 0.000e+00, 0.000e+00,
        1.000e+00],
       [1.000e+00, 1.836e+03, 2.000e+00, …, 0.000e+00, 0.000e+00,
        1.000e+00]])
```

[112]:

```
# scale the feature matrix using the fit_transform() method of the scaler object
X_scaled = scaler.fit_transform(X)

# assign the scaled feature matrix to the original variable name 'X'
X = X_scaled
```

[113]:

```
# create a 1D NumPy array of zeros with length 12
theta = np.zeros(12)

# reshape the 1D array to a column vector using np.reshape
theta = np.reshape(theta, (12,1))

# display the column vector
theta
```

[113]:

```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[114]:

```
# create a 1D NumPy array 'Y_t' from the 'price' column of the training set␣
 ↪'train'
Y_t = np.array(train.price)

# create a copy of 'Y_t' to prevent changing the original data
Y = Y_t.copy()

# reshape 'Y' to a column vector using np.reshape
Y = np.reshape(Y, (436,1))

# create an instance of the MinMaxScaler class for scaling the target variable␣
 ↪to a range of [0, 1]
scaler = sklearn.preprocessing.MinMaxScaler()

# scale the target variable using the fit_transform() method of the scaler␣
 ↪object
Y_scaled = scaler.fit_transform(Y)

# assign the scaled target variable to the original variable name 'Y'
```

```
Y = Y_scaled
```

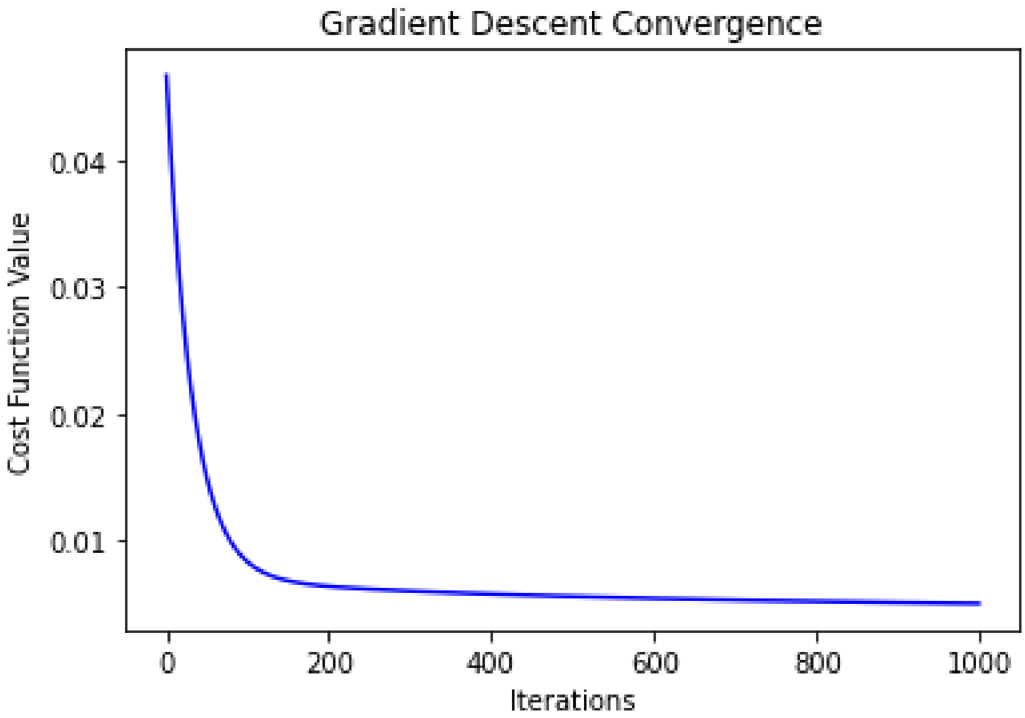
[115]:

```
# create a NumPy array 'X_T' containing the transpose of the feature matrix 'X'
X_T = np.array(X.T)

# retrieve the number of rows 'm' and the number of columns 'n' from the␣
 ↪feature matrix 'X'
m, n = X.shape

# display the values of 'm' and 'n'
print("Number of training examples (m): ", m)
print("Number of features (n): ", n)

# set the number of iterations for gradient descent
iterations = 1000

# create a counter variable 'count' and a NumPy array 'j' to store the cost␣
 ↪function values for each iteration
count = 0
j = np.zeros(shape=(iterations, 1), dtype=float)

# display the shape of the 'j' array
print("Shape of 'j' array: ", j.shape)
```

```
Number of training examples (m):  436
Number of features (n):  12
Shape of 'j' array:  (1000, 1)
```

[116]:

```
# set the initial iteration count to zero
count = 0

# create a NumPy array 'j' to store the cost function values for each iteration
j = np.zeros(shape=(iterations, 1), dtype=float)

# perform gradient descent for the specified number of iterations
while count < iterations:

    # calculate the predicted values 'h' using the current parameters 'theta'
    h = X.dot(theta)

    # calculate the cost function value 'j' using the current parameters 'theta'
    j[count] = (1/(2*m)) * np.sum((h-Y)**2)

    # calculate the gradient of the cost function with respect to 'theta'
    grad = (1/m) * X_T.dot(h-Y)
    # update the parameters 'theta' using the learning rate 'alpha' and the␣
 ↪gradient 'grad'
    alpha = 0.01
    theta = theta - alpha * grad

    # increment the iteration count
    count += 1

# plot the cost function values over the iterations
plt.plot(j,'b-')
plt.xlabel('Iterations')
plt.ylabel('Cost Function Value')
plt.title('Gradient Descent Convergence')
plt.show()
```



[117]:

```
# extract the test set features into NumPy arrays
N_X1_t = np.array(test.area)
N_X2_t = np.array(test.bedrooms)
N_X3_t = np.array(test.bathrooms)
N_X4_t = np.array(test.stories)
N_X5_t = np.array(test.parking)
N_X6_t = np.array(test.mainroad)
N_X7_t = np.array(test.guestroom)
```

```
N_X8_t = np.array(test.hotwaterheating)
N_X9_t = np.array(test.airconditioning)
N_X10_t = np.array(test.prefarea)
N_X11_t = np.array(test.basement)
N_X0_t = np.ones(109)
```

[118]:
```
# stack the test set features into a design matrix
N_X = np.vstack([N_X0_t,N_X1_t,N_X2_t,N_X3_t,N_X4_t,N_X5_t,␣
 ↪N_X6_t,N_X7_t,N_X8_t,N_X9_t,N_X10_t,N_X11_t])
N_X_T = N_X.T
N_X = np.array(N_X_T)
N_x = scaler.fit_transform(N_X)
N_X = N_x
N_X.shape
```

[118]:
```
(109, 12)
```

[119]:
```
# initialize the parameters for the test set
N_theta = np.array([0.,0.,0.,0.,0,0.,0.,0.,0.,0.,0,0.])
N_theta = N_theta.reshape(12,1)
N_theta
```

[119]:
```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[120]:
```
# initialize the target variable for the test set
N_Y_t = np.array(test.price)
N_Y = N_Y_t
N_Y = N_Y_t.reshape(109,1)
N_y = scaler.fit_transform(N_Y)
N_Y=N_y
N_Y.shape
```

[120]:
```
(109, 1)
```

[121]:
```
N_X_T = np.array(N_X.T)
m,n = N_X.shape
```

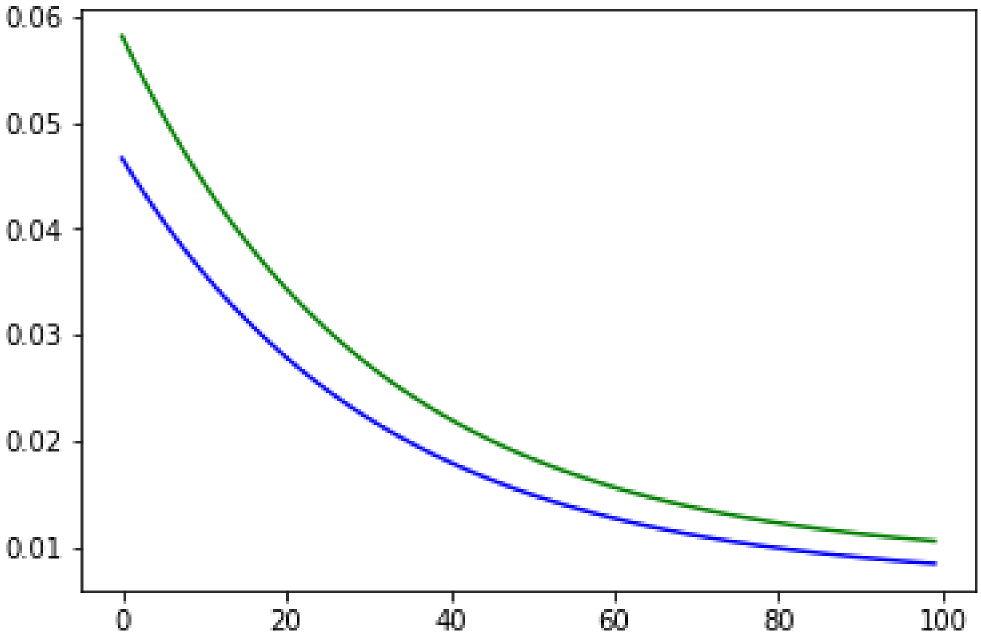
```
m,n
```

[121]:
```
(109, 12)
```

[122]:
```
iterations = 1000
count=0
N_j = np.zeros(shape=(iterations, 1), dtype=float)

while(count < iterations):

    N_h_t = N_X.dot(N_theta)
    N_h = np.array(N_h_t)


    N_j[count]= (1/(2*m))*np.sum((N_h - N_Y)**2)

    grad_t = N_X_T.dot(N_h-N_Y)
    grad = grad_t*(1/m)

    N_theta = N_theta - 0.01*(grad)

    count += 1
```

[127]:
```
plt.plot(N_j[:100],'g-')
plt.plot(j[:100],'b-')
```

[127]:
```
[<matplotlib.lines.Line2D at 0x7faee47df520>]
```



hw1-3b-standardization

February 20, 2023

[1]:

```
import numpy as np # import numpy library
import pandas as pd # import pandas library
import matplotlib.pyplot as plt # import matplotlib library
from sklearn import preprocessing
```

[2]:

```
df = pd.read_csv("/content/sample_data/Housing.csv")

# display DataFrame
df
```

[2]:

```
        price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0    13300000  7420         4          2        3      yes        no       no
1    12250000  8960         4          4        4      yes        no       no
2    12250000  9960         3          2        2      yes        no      yes
3    12215000  7500         4          2        2      yes        no      yes
4    11410000  7420         4          1        2      yes       yes      yes
..        …     …       …        …      …      …       …      …
540   1820000  3000         2          1        1      yes        no      yes
541   1767150  2400         3          1        1       no        no       no
542   1750000  3620         2          1        1      yes        no       no
543   1750000  2910         3          1        1       no        no       no
544   1750000  3850         3          1        2      yes        no       no

    hotwaterheating airconditioning  parking prefarea furnishingstatus
0                no             yes        2      yes        furnished
1                no             yes        3       no        furnished
2                no              no        2      yes   semi-furnished
3                no             yes        3      yes        furnished
4                no             yes        2       no        furnished
..              …             …      …      …              …
540              no              no        2       no      unfurnished
541              no              no        0       no   semi-furnished
542              no              no        0       no      unfurnished
543              no              no        0       no        furnished
544              no              no        0       no      unfurnished

[545 rows x 13 columns]
```

[3]:
```
df = df.replace(to_replace=['yes', 'no'], value=[1, 0])
df
```

[3]:
```
        price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0    13300000  7420         4          2        3         1          0
1    12250000  8960         4          4        4         1          0
2    12250000  9960         3          2        2         1          0
3    12215000  7500         4          2        2         1          0
4    11410000  7420         4          1        2         1          1
..        …     …         …          …        …         …
540   1820000  3000         2          1        1         1          0
541   1767150  2400         3          1        1         0          0
542   1750000  3620         2          1        1         1          0
543   1750000  2910         3          1        1         0          0
544   1750000  3850         3          1        2         1          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
0           0                0                1        2         1
1           0                0                1        3         0
2           1                0                0        2         1
3           1                0                1        3         1
4           1                0                1        2         0
..        …                …                …       …         …
540         1                0                0        2         0
541         0                0                0        0         0
542         0                0                0        0         0
543         0                0                0        0         0
544         0                0                0        0         0

    furnishingstatus
0          furnished
1          furnished
2     semi-furnished
3          furnished
4          furnished
..               …
540      unfurnished
541   semi-furnished
542      unfurnished
543        furnished
544      unfurnished

[545 rows x 13 columns]
```

[4]:
```
# create a training set by randomly selecting 80% of the rows from the DataFrame
train = df.sample(frac=0.8, random_state=1)
```

```
# create a test set by dropping the rows in the training set from the DataFrame
test = df.drop(train.index)
```

[5]:

```
train
```

[5]:

```
       price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
62   7070000   6240         4          2        2         1          0
247  4550000   8400         4          1        4         1          0
142  5600000  10500         4          2        2         1          0
107  6125000   6420         3          1        3         1          0
483  2940000   6615         3          1        2         1          0
..       …      …        …          …        …         …
359  3710000   3600         3          1        1         1          0
36   8043000   7482         3          2        3         1          0
30   8400000   7475         3          2        4         1          0
20   8750000   4320         3          1        2         1          0
527  2275000   1836         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
62          0                0                1        1         0
247         0                0                0        3         0
142         0                0                0        1         0
107         1                0                0        0         1
483         0                0                0        0         0
..        …              …               …      …       …
359         0                0                0        1         0
36          0                1                0        1         1
30          0                0                1        2         0
20          1                1                0        2         0
527         1                0                0        0         0

    furnishingstatus
62         furnished
247      unfurnished
142   semi-furnished
107      unfurnished
483   semi-furnished
..               …
359      unfurnished
36         furnished
30       unfurnished
20    semi-furnished
527   semi-furnished

[436 rows x 13 columns]
```

[6]:

```
test
```

[6]:
```
        price   area  bedrooms  bathrooms  stories  mainroad  guestroom  \
2    12250000   9960         3          2        2         1          0
3    12215000   7500         4          2        2         1          0
7    10150000  16200         5          3        2         1          0
15    9100000   6000         4          1        2         1          0
22    8645000   8050         3          1        1         1          1
..        …      …       …         …        …        …         …
508   2590000   4400         2          1        1         1          0
513   2485000   4400         3          1        2         1          0
520   2450000   7700         2          1        1         1          0
537   1890000   1700         3          1        2         1          0
539   1855000   2990         2          1        1         0          0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
2           1                0                0        2         1
3           1                0                1        3         1
7           0                0                0        0         0
15          1                0                0        2         0
22          1                0                1        1         0
..        …              …               …      …        …
508         0                0                0        0         0
513         0                0                0        0         0
520         0                0                0        0         0
537         0                0                0        0         0
539         0                0                0        1         0

    furnishingstatus
2     semi-furnished
3          furnished
7        unfurnished
15    semi-furnished
22         furnished
..               …
508      unfurnished
513      unfurnished
520      unfurnished
537      unfurnished
539      unfurnished

[109 rows x 13 columns]
```

[7]:
```
# select specific columns for the training set
train = train[['price','area','bedrooms','bathrooms','stories','parking',␣
 ↪'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',␣
 ↪'prefarea']]
# select specific columns for the test set
```

```
test = test[['price','area','bedrooms','bathrooms','stories','parking',␣
 ↪'mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning',␣
 ↪'prefarea']]
```

[8]:
```
scaler = preprocessing.StandardScaler()
```

[9]:
```
train
```

[9]:
```
       price   area  bedrooms  bathrooms  stories  parking  mainroad  \
62   7070000   6240         4          2        2        1         1
247  4550000   8400         4          1        4        3         1
142  5600000  10500         4          2        2        1         1
107  6125000   6420         3          1        3        0         1
483  2940000   6615         3          1        2        0         1
..       …      …       …        …       …       …        …
359  3710000   3600         3          1        1        1         1
36   8043000   7482         3          2        3        1         1
30   8400000   7475         3          2        4        2         1
20   8750000   4320         3          1        2        2         1
527  2275000   1836         2          1        1        0         0

     guestroom  basement  hotwaterheating  airconditioning  prefarea
62           0         0                0                1         0
247          0         0                0                0         0
142          0         0                0                0         0
107          0         1                0                0         1
483          0         0                0                0         0
..         …        …              …              …       …
359          0         0                0                0         0
36           0         0                1                0         1
30           0         0                0                1         0
20           0         1                1                0         0
527          0         1                0                0         0

[436 rows x 12 columns]
```

[10]:
```
# create a NumPy array of the 'area' column from the training set
X1_t = np.array(train['area'])

# display the NumPy array
X1_t
```

[10]:
```
array([ 6240,  8400, 10500,  6420,  6615,  3600,  3240,  6600,  2700,
        5000,  2650,  4775,  4800,  3700,  7700,  7420,  4280,  6000,
        6600,  3649,  3420,  5500,  3630,  3180,  3600,  8400,  3000,
        8880,  5750,  2145,  6360,  6525,  1950,  5850,  8372,  2870,
        4990,  2684,  5200,  6321,  4960,  3480,  3210,  4950,  6840,
        4350,  5850,  4410,  2500,  3850,  3180,  3162,  3500,  4340,
        6440,  5010,  3000,  4920,  3760,  3816,  6000,  7000,  3640,
        4080,  4160,  2910,  6060,  3000,  2787,  4815,  4785,  6600,
        5300,  3600,  6000,  2176,  3000,  7420,  7020,  3480,  5960,
        3510,  6420,  6450,  6210,  4500,  3000,  3180,  5700,  3520,
        4040,  5800,  2800,  6480,  4960,  4260,  7500,  5880, 10500,
        4500,  3850,  8500,  3120,  3990,  4095,  4800, 13200,  7770,
        6100,  4075,  6550,  4100,  4370,  3180,  7350,  3510,  3640,
        5500,  8250,  6600,  8250,  2475,  3850,  4500,  3720,  4360,
       10240,  5500,  3970,  3450,  3850,  5500,  3520,  2145,  6600,
        3640,  3986,  2953,  8250,  4130,  8580,  6000,  3500,  5885,
        7680,  2430,  3150,  6450,  8100,  5500,  1650,  3040,  4079,
        2747,  4600,  2325,  7231,  3520,  2145,  3450,  3620,  4000,
        6000,  6000,  4500,  3540,  7200,  3120,  4000,  2015,  4040,
        8000,  2787,  3512,  3420,  6060,  4500,  6360,  5450,  8250,
        3960,  7410, 10360,  3630,  6020,  4100,  6254,  4500,  4560,
        6710,  3500,  8880,  3600,  7152,  6000,  4040,  4000,  4040,
        5360,  6600,  3800,  3960,  4900,  3480,  3584,  2275,  4000,
        6500, 10500,  8960,  3290,  8875,  8580,  3450,  6600,  2800,
        5640,  3745, 10269,  6100, 12090,  5880,  6750,  6000,  5320,
        4000,  4040, 15600,  3090,  3970,  5450,  4770,  4095,  6000,
        6540,  6550,  4320,  3100,  4050,  3650,  3850,  5600,  2817,
        4510,  3000,  4995, 11410,  3000,  4840,  3600,  4000,  3500,
        7800,  5300,  4840,  3000,  3480,  2970,  5828,  3800,  4040,
       10700,  7320,  5000,  6325,  2880,  4300,  3150,  4000,  9500,
        4500,  3420,  3180,  2145,  5400,  3630,  6750,  4820,  5136,
        4120,  6825,  4600,  6650,  5800,  5720,  5000,  4352,  3300,
        2160,  5900,  3000,  4500,  3350,  5400,  4600,  9800,  3630,
        2610,  9667,  3635,  4000,  3180,  3630,  6600,  2610,  4960,
        5150,  6000,  3640,  2910,  3650,  3450,  4032,  7980,  1905,
        6000,  3360,  9620,  1950, 12900,  3240,  4320,  6540,  6000,
        7440,  3760,  8100,  4880,  6000,  2000,  5200,  4050,  9166,
        7950,  5500,  2700,  6000,  6900,  3500,  5076,  5985,  4300,
        8050,  5320,  5960,  7000,  7260,  6360,  3000,  3460, 12944,
        3880,  2400,  4080,  6000,  4500,  6050,  7000,  3930,  4600,
        7155,  4100,  2400,  3460,  4632,  4200,  4640,  8800,  3000,
        6300,  7000,  7000,  6900,  3420,  3264,  2640,  3150,  4320,
        6862, 11440,  4992,  3069,  3185,  3750,  5300,  7200,  6400,
        6800,  3400,  6420,  3792,  5500,  4600,  6800,  6000,  8520,
        6480,  8150,  5948,  3185,  5830,  3410,  3000,  8400,  6350,
        8100,  4800,  2856,  3185,  3780,  3640,  6000,  6000,  4800,
        5800,  6360,  4120,  5400,  2850,  5400,  2145,  4500,  3240,
       13200,  3900,  9000,  4646,  3840,  9000,  3520,  3640,  3600,
        7482,  7475,  4320,  1836])
```

[11]:

```
X2_t = np.array(train.bedrooms)
X3_t = np.array(train.bathrooms)
```

```
X4_t = np.array(train.stories)
X5_t = np.array(train.parking)
X6_t = np.array(train.mainroad)
X7_t = np.array(train.guestroom)
X8_t = np.array(train.hotwaterheating)
X9_t = np.array(train.airconditioning)
X10_t = np.array(train.prefarea)
X11_t = np.array(train.basement)
X0_t= np.ones(436)
```

[12]:

```
X = np.vstack([X0_t,X1_t,X2_t,X3_t,X4_t,X5_t,X6_t,X7_t,X8_t,X9_t,X10_t,X11_t])
X = X.T
X = np.array(X)
X
```

[12]:

```
array([[1.000e+00, 6.240e+03, 4.000e+00, …, 1.000e+00, 0.000e+00,
        0.000e+00],
       [1.000e+00, 8.400e+03, 4.000e+00, …, 0.000e+00, 0.000e+00,
        0.000e+00],
       [1.000e+00, 1.050e+04, 4.000e+00, …, 0.000e+00, 0.000e+00,
        0.000e+00],
       …,
       [1.000e+00, 7.475e+03, 3.000e+00, …, 1.000e+00, 0.000e+00,
        0.000e+00],
       [1.000e+00, 4.320e+03, 3.000e+00, …, 0.000e+00, 0.000e+00,
        1.000e+00],
       [1.000e+00, 1.836e+03, 2.000e+00, …, 0.000e+00, 0.000e+00,
        1.000e+00]])
```

[13]:

```
# scale the feature matrix using the fit_transform() method of the scaler object
X_scaled = scaler.fit_transform(X)

# assign the scaled feature matrix to the original variable name 'X'
X = X_scaled
```

[14]:

```
# create a 1D NumPy array of zeros with length 12
theta = np.zeros(12)

# reshape the 1D array to a column vector using np.reshape
theta = np.reshape(theta, (12,1))

# display the column vector
theta
```

[14]:

```
array([[0.],
       [0.],
       [0.],
        [0.],
        [0.],
        [0.],
        [0.],
        [0.],
        [0.],
        [0.],
        [0.],
        [0.]])
```

[15]:

```
# create a 1D NumPy array 'Y_t' from the 'price' column of the training set␣
 ↪'train'
#h = np.matmul(X,theta)
Y_t = np.array(train.price)
Y = Y_t
Y = Y_t.reshape(436,1)
y = scaler.fit_transform(Y)
Y=y
```

[16]:

```
# create a NumPy array 'X_T' containing the transpose of the feature matrix 'X'
X_T = np.array(X.T)

# retrieve the number of rows 'm' and the number of columns 'n' from the␣
 ↪feature matrix 'X'
m, n = X.shape

# display the values of 'm' and 'n'
print("Number of training examples (m): ", m)
print("Number of features (n): ", n)

# set the number of iterations for gradient descent
iterations = 1000

# create a counter variable 'count' and a NumPy array 'j' to store the cost␣
 ↪function values for each iteration
count = 0
j = np.zeros(shape=(iterations, 1), dtype=float)

# display the shape of the 'j' array
print("Shape of 'j' array: ", j.shape)
```

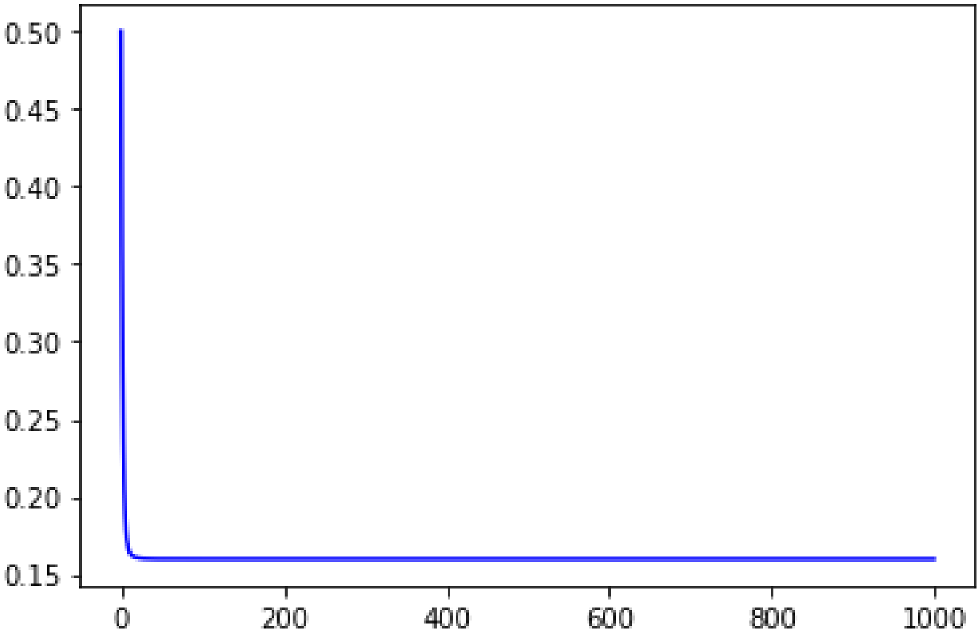
```
Number of training examples (m):  436
Number of features (n):  12
Shape of 'j' array:  (1000, 1)
```

[17]:

```
# Initialize variables
iterations = 1000
```

```
count = 0
alpha = 0.1
lambda_ = 0.001
j = np.zeros(iterations)

# Perform gradient descent
while count < iterations:
    h_t = X.dot(theta)
    h = np.array(h_t, float)

    j[count] = (1/(2*m)) * np.sum((h - Y)**2) + (lambda_ / (2*m)) * np.sum(theta**2)

    grad_t = X_T.dot(h - Y)
    grad = grad_t * (1/m)

    theta = theta * (1 - alpha * (lambda_ / m)) - alpha * grad

    count += 1

# Plot the cost function
plt.plot(j, 'b-')
```

```
[17]: [<matplotlib.lines.Line2D at 0x7f03efeffd60>]
```



[18]:

```
# extract the test set features into NumPy arrays
N_X1_t = np.array(test.area)
N_X2_t = np.array(test.bedrooms)
N_X3_t = np.array(test.bathrooms)
N_X4_t = np.array(test.stories)
N_X5_t = np.array(test.parking)
N_X6_t = np.array(test.mainroad)
N_X7_t = np.array(test.guestroom)
N_X8_t = np.array(test.hotwaterheating)
N_X9_t = np.array(test.airconditioning)
N_X10_t = np.array(test.prefarea)
N_X11_t = np.array(test.basement)
N_X0_t = np.ones(109)
```

[19]:

```
# stack the test set features into a design matrix
N_X = np.vstack([N_X0_t,N_X1_t,N_X2_t,N_X3_t,N_X4_t,N_X5_t,␣
 ↪N_X6_t,N_X7_t,N_X8_t,N_X9_t,N_X10_t,N_X11_t])
N_X_T = N_X.T
N_X = np.array(N_X_T)
N_x = scaler.fit_transform(N_X)
N_X = N_x
N_X.shape
```

[19]:

```
(109, 12)
```

[20]:

```
# initialize the parameters for the test set
N_theta = np.array([0.,0.,0.,0.,0,0.,0.,0.,0.,0.,0,0.])
N_theta = N_theta.reshape(12,1)
N_theta
```

[20]:

```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]])
```

[21]:

```
# initialize the target variable for the test set
N_Y_t = np.array(test.price)
N_Y = N_Y_t
N_Y = N_Y_t.reshape(109,1)
```

```
N_y = scaler.fit_transform(N_Y)
N_Y=N_y
N_Y.shape
```

```
[21]: (109, 1)
```

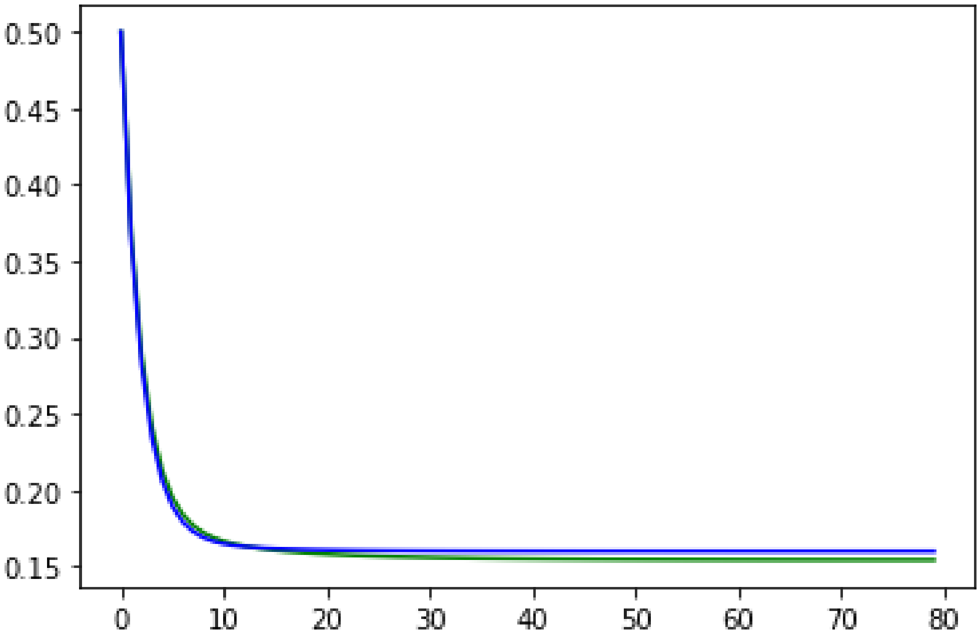
```
[22]: N_X_T = np.array(N_X.T)
      m,n = N_X.shape
      m,n
```

```
[22]: (109, 12)
```

```
[24]: iterations = 2500
      count=0
      N_j = np.zeros(shape=(iterations, 1), dtype=float)

      while(count < iterations):

          N_h_t = N_X.dot(N_theta)
          N_h = np.array(N_h_t)


          N_j[count]= (1/(2*m))*np.sum((N_h - N_Y)**2)+(0.001)*(1/(2*m))*np.
       ↪sum((N_theta)**2)

          grad_t = N_X_T.dot(N_h-N_Y)
          grad = grad_t*(1/m)

          N_theta = N_theta*(1-0.1*(0.001/m)) - 0.1*(grad)

          count += 1
```

```
[26]: plt.plot(N_j[:80],'g-')
      plt.plot(j[:80],'b-')
```

```
[26]: [<matplotlib.lines.Line2D at 0x7f03ef903eb0>]
```


0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
10
20
30
40
50
60
70
80