Name: YASH DARJI Roll No.: 19BCE043 Course : Machine Learning Course Code : 2CS501 Practical : 3

Here I am using Gradient Decsent method.

Boston Housing with Gradient Decsent Regression

```
import numpy as np
from sklearn import datasets,metrics
from sklearn.preprocessing import StandardScaler
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

X,y = datasets.load_boston(return_X_y=True)
print(X.shape)
#BOSTON.head(5)
```

```
     (506, 13)
```

```
#Converting dataset  into dataframe
from sklearn.datasets import load_boston
boston_dataset = load_boston()
boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
boston.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90

```
x_training_temp=X[0:400,:]
x_train=np.zeros((x_training_temp.shape[0],x_training_temp.shape[1]+1))
x_train[:,0]=np.ones((x_training_temp.shape[0]))
#print(x_train)

x_train[:,1:]=x_training_temp
print("Type of x_train:",type(x_train))
print("Shape of x_train:",x_train.shape)
#print(x_train)

y_train=y[0:400]
x_test_temp=X[400:506,:]
```

```
x_test_temp=x[400:506,:]
x_test=np.zeros((x_test_temp.shape[0],x_test_temp.shape[1]+1))
x_test[:,1:]=x_test_temp


print("Type of x_train:",type(x_test))
print("Shape of x_train:",x_test.shape)
y_test=y[400:506]
print(x_train)




scaler=StandardScaler()
scaler.fit(x_train[:,1:])
x_train[:,1:]=scaler.transform(x_train[:,1:])
x_test[:,1:]=scaler.transform(x_test[:,1:])
print(x_train)
print(x_train.T)
```

```
Type of x_train: <class 'numpy.ndarray'>
Shape of x_train: (400, 14)
Type of x_train: <class 'numpy.ndarray'>
Shape of x_train: (106, 14)
[[1.00000e+00 6.32000e-03 1.80000e+01 ... 1.53000e+01 3.96900e+02
  4.98000e+00]
 [1.00000e+00 2.73100e-02 0.00000e+00 ... 1.78000e+01 3.96900e+02
  9.14000e+00]
 [1.00000e+00 2.72900e-02 0.00000e+00 ... 1.78000e+01 3.92830e+02
  4.03000e+00]
 ...
 [1.00000e+00 7.67202e+00 0.00000e+00 ... 2.02000e+01 3.93100e+02
  1.99200e+01]
 [1.00000e+00 3.83518e+01 0.00000e+00 ... 2.02000e+01 3.96900e+02
  3.05900e+01]
 [1.00000e+00 9.91655e+00 0.00000e+00 ... 2.02000e+01 3.38160e+02
  2.99700e+01]]
[[ 1.         -0.29171468  0.14290806 ... -1.21945879  0.41941166
  -0.92732759]
 [ 1.         -0.28820481 -0.56670438 ... -0.09220903  0.41941166
  -0.3121236 ]
 [ 1.         -0.28820816 -0.56670438 ... -0.09220903  0.3192679
  -1.06781889]
 ...
 [ 1.          0.99011231 -0.56670438 ...  0.98995074  0.32591134
   1.2820829 ]
 [ 1.          6.12025956 -0.56670438 ...  0.98995074  0.41941166
   2.86002199]
 [ 1.          1.36543343 -0.56670438 ...  0.98995074 -1.02590637
   2.76833293]]
[[ 1.          1.          1.         ...  1.          1.
   1.        ]
 [-0.29171468 -0.28820481 -0.28820816 ...  0.99011231  6.12025956
   1.36543343]
 [ 0.14290806 -0.56670438 -0.56670438 ... -0.56670438 -0.56670438
  -0.56670438]
 ...
 [-1.21945879 -0.09220903 -0.09220903 ...  0.98995074  0.98995074
   0.98995074]
 [ 0.41941166  0.41941166  0.3192679  ...  0.32591134  0.41941166
  -1.02590637]
 [-0.92732759 -0.3121236  -1.06781889 ...  1.2820829   2.86002199
   2.76833293]]
```

```
theta=np.random.uniform(0,1,size=(x_train.shape[1]))
print("Type of theta:",type(theta))
print("Shape of theta:",theta.shape)
iterations=1000
alpha=0.01
m=x_train.shape[0]
c=x_train.shape[1]

for i in range(iterations):
  update=np.zeros(x_train.shape[1])
  y_pred=np.dot(x_train,theta)
  error=y_pred - y_train
  for j in range(c):
    update[j]=np.sum(error*(x_train.T)[j])
  theta= theta - (1/m)*(alpha)*update


print('THETA',theta)
print('THETA shape',theta.shape)

pred=np.dot(x_test,theta)
print("MAE:",metrics.mean_absolute_error(y_true=y_test,y_pred=pred))
print("MSE:",metrics.mean_squared_error(y_true=y_test,y_pred=pred))
```

```
Type of theta: <class 'numpy.ndarray'>
Shape of theta: (14,)
THETA [24.33346823 -1.02907295  0.9096029   0.04224119  0.54797174 -1.15468776
  3.75555767 -0.05072687 -2.49553979  2.21905903 -1.25013164 -1.65635589
  0.04762351 -3.44836114]
THETA shape (14,)
MAE: 21.10820696375072
MSE: 468.80915517883045
```

● ✕