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

θ^=(XT·X)^−1·(XT·y)

```
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
from numpy.linalg import inv , pinv ,LinAlgError
```

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

print(X)
```

```
(506, 13)
[[6.3200e-03 1.8000e+01 2.3100e+00 ... 1.5300e+01 3.9690e+02 4.9800e+00]
 [2.7310e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01 3.9690e+02 9.1400e+00]
 [2.7290e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01 3.9283e+02 4.0300e+00]
 ...
 [6.0760e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9690e+02 5.6400e+00]
 [1.0959e-01 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9345e+02 6.4800e+00]
 [4.7410e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9690e+02 7.8800e+00]]
```

```
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_training:",type(x_train))
print("Shape of x_training:",x_train.shape)
print(x_train)
```

```
[[1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 ...
 [1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]]
Type of x_training: <class 'numpy.ndarray'>
Shape of x_training: (400, 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]]
```

```
y_train=y[0:400]
print(y_train)

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)
```

```
[24.  21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 15.  18.9 21.7 20.4
 18.2 19.9 23.1 17.5 20.2 18.2 13.6 19.6 15.2 14.5 15.6 13.9 16.6 14.8
 18.4 21.  12.7 14.5 13.2 13.1 13.5 18.9 20.  21.  24.7 30.8 34.9 26.6
 25.3 24.7 21.2 19.3 20.  16.6 14.4 19.4 19.7 20.5 25.  23.4 18.9 35.4
 24.7 31.6 23.3 19.6 18.7 16.  22.2 25.  33.  23.5 19.4 22.  17.4 20.9
 24.2 21.7 22.8 23.4 24.1 21.4 20.  20.8 21.2 20.3 28.  23.9 24.8 22.9
 23.9 26.6 22.5 22.2 23.6 28.7 22.6 22.  22.9 25.  20.6 28.4 21.4 38.7
 43.8 33.2 27.5 26.5 18.6 19.3 20.1 19.5 19.5 20.4 19.8 19.4 21.7 22.8
 18.8 18.7 18.5 18.3 21.2 19.2 20.4 19.3 22.  20.3 20.5 17.3 18.8 21.4
 15.7 16.2 18.  14.3 19.2 19.6 23.  18.4 15.6 18.1 17.4 17.1 13.3 17.8
 14.  14.4 13.4 15.6 11.8 13.8 15.6 14.6 17.8 15.4 21.5 19.6 15.3 19.4
 17.  15.6 13.1 41.3 24.3 23.3 27.  50.  50.  50.  22.7 25.  50.  23.8
 23.8 22.3 17.4 19.1 23.1 23.6 22.6 29.4 23.2 24.6 29.9 37.2 39.8 36.2
 37.9 32.5 26.4 29.6 50.  32.  29.8 34.9 37.  30.5 36.4 31.1 29.1 50.
 33.3 30.3 34.6 34.9 32.9 24.1 42.3 48.5 50.  22.6 24.4 22.5 24.4 20.
 21.7 19.3 22.4 28.1 23.7 25.  23.3 28.7 21.5 23.  26.7 21.7 27.5 30.1
 44.8 50.  37.6 31.6 46.7 31.5 24.3 31.7 41.7 48.3 29.  24.  25.1 31.5
 23.7 23.3 22.  20.1 22.2 23.7 17.6 18.5 24.3 20.5 24.5 26.2 24.4 24.8
 29.6 42.8 21.9 20.9 44.  50.  36.  30.1 33.8 43.1 48.8 31.  36.5 22.8
 30.7 50.  43.5 20.7 21.1 25.2 24.4 35.2 32.4 32.  33.2 33.1 29.1 35.1
 45.4 35.4 46.  50.  32.2 22.  20.1 23.2 22.3 24.8 28.5 37.3 27.9 23.9
 21.7 28.6 27.1 20.3 22.5 29.  24.8 22.  26.4 33.1 36.1 28.4 33.4 28.2
 22.8 20.3 16.1 22.1 19.4 21.6 23.8 16.2 17.8 19.8 23.1 21.  23.8 23.1
 20.4 18.5 25.  24.6 23.  22.2 19.3 22.6 19.8 17.1 19.4 22.2 20.7 21.1
 19.5 18.5 20.6 19.  18.7 32.7 16.5 23.9 31.2 17.5 17.2 23.1 24.5 26.6
 22.9 24.1 18.6 30.1 18.2 20.6 17.8 21.7 22.7 22.6 25.  19.9 20.8 16.8
 21.9 27.5 21.9 23.1 50.  50.  50.  50.  50.  13.8 13.8 15.  13.9 13.3
 13.1 10.2 10.4 10.9 11.3 12.3  8.8  7.2 10.5  7.4 10.2 11.5 15.1 23.2
  9.7 13.8 12.7 13.1 12.5  8.5  5.   6.3]
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]]
```

we are adding a column of ones to make it suitable for dot product between the two matrices.

You have y = w0 + w1*x. In linear algebra, it can be written like this :

y = [x]* w0 w1

because the two matrices do not have the same internal size (the size of [x] is n by 1 and the size of [w0 w1] (transpose) is 2 by 1) so if we want to calculate dot product, we have to add an extra column of ones and the operation can be written as:

y = [1 x]* w0 w1

```
theta = np.zeros(x_train.shape[1])

try:
  xtxi=inv(np.dot(x_train.T,x_train))
except:
  xtxi=pinv(np.dot(x_train.T,x_train))

xty=np.dot(x_train.T,y_train)
theta = np.dot(xtxi,xty)



print("THETA SHAPE",theta.shape)
print("THETA",theta)

print(x_test.shape)
```

```
p=np.dot(theta, x_test.T)

print(p.shape,"\n\n\n",y_test.shape)

print("MAE:",metrics.mean_absolute_error(y_true=y_test,y_pred=p))
print("MSE:",metrics.mean_squared_error(y_true=y_test,y_pred=p))
```

```
THETA SHAPE (14,)
THETA SHAPE (14,)
THETA [ 2.86725996e+01 -1.91246374e-01  4.42289967e-02  5.52207977e-02
  1.71631351e+00 -1.49957220e+01  4.88773025e+00  2.60921031e-03
 -1.29480799e+00  4.84787214e-01 -1.54006673e-02 -8.08795026e-01
 -1.29230427e-03 -5.17953791e-01]
(106, 14)
(106,)


 (106,)
MAE: 25.089692427710634
MSE: 654.5492209773108
```