In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

#1.read in the values
data = pd.read_csv('x06Simple.csv')
y = data.to_numpy()

#2.Randomizes the data
np.random.seed(0)
np.random.shuffle(y)
size = np.size(y[:,1])

#3.Selects the first 2/3 (round up) of the data for training and the remaining for testing
x = math.ceil(size*(2/3))
Yan = np.array(y[0:x,[3]]) #2/3 train data
yanTest = np.array(y[x:,[3]]) #1/3 test data
mat = np.array(y[:,1:3])
one = np.ones((size,1))
mat = np.concatenate((one, mat), axis=1)
mattrain = mat[0:x,:]
mat_test = mat[x:,:]

#4.Standardizes the data (except for the last column of course) using the training data
mean2 = np.mean(mattrain ,axis=0)
std2 = np.std(mattrain, axis=0,ddof=1)
mattrain[:,[1,2]] = (mattrain[:,[1,2]]-mean2[1:3])/std2[1:3]
mat_test[:,[1,2]] = (mat_test[:,[1,2]]-mean2[1:3])/std2[1:3]
mat2= mattrain

#5.Computes the closed-form solution of linear regression
theta = np.linalg.inv(mat2.T @ (mat2)) @ (mat2.T @ Yan)
print("theta values")
print(theta)

#6.Applies the solution to the testing samples
y1 = theta[0] + mat_test[:,1]*theta[1]+mat_test[:,2]*theta[2]
```

```
#7.Computes the root mean squared error (RMSE)
error = np.sqrt(mean_squared_error(y1,yanTest))
print("RMSE")
print(error)
y = theta[0] + theta[1]*mat_test[1]+theta[2]*mat_test[2]
print("Y = {0}+{1}*x1-{2}*x2".format(theta[0],theta[1],theta[2]))
```

```
theta values
[[3275.66666667]
 [1097.60312694]
 [-259.32789051]]
RMSE
601.930289499697
Y = [3275.66666667]+[1097.60312694]*x1-[-259.32789051]*x2
```

```
In [ ]:
```