Assignment 3

- Sambit Tarai
- s4596952

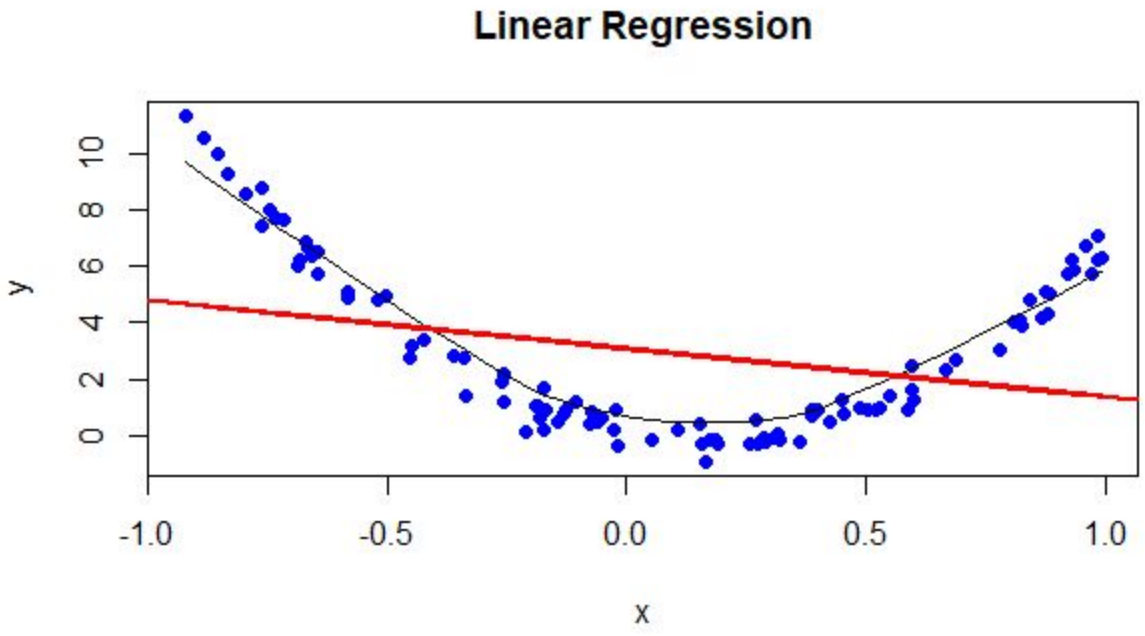
Consider the two variables in the dataset Assign.csv. We are interested in predicting the second variable Y given the first variable X.

1. Fit a Linear Regression Model to the data. Show the data scatter plot on the same figure with the values predicted by the linear model.

- **Code:**

```
data1 = read.csv('Assign3.csv')
attach(data1)
lm.fit = lm(y~x, data=data1)
z = predict(lm.fit, data=data1)
scatter.smooth(x=x,y=y, col="blue", pch=19, main="Linear Regression")
abline(lm.fit, col="red", lwd=2)
```

- **Here, Blue points - Actual data points & Red Line - Predicted values**

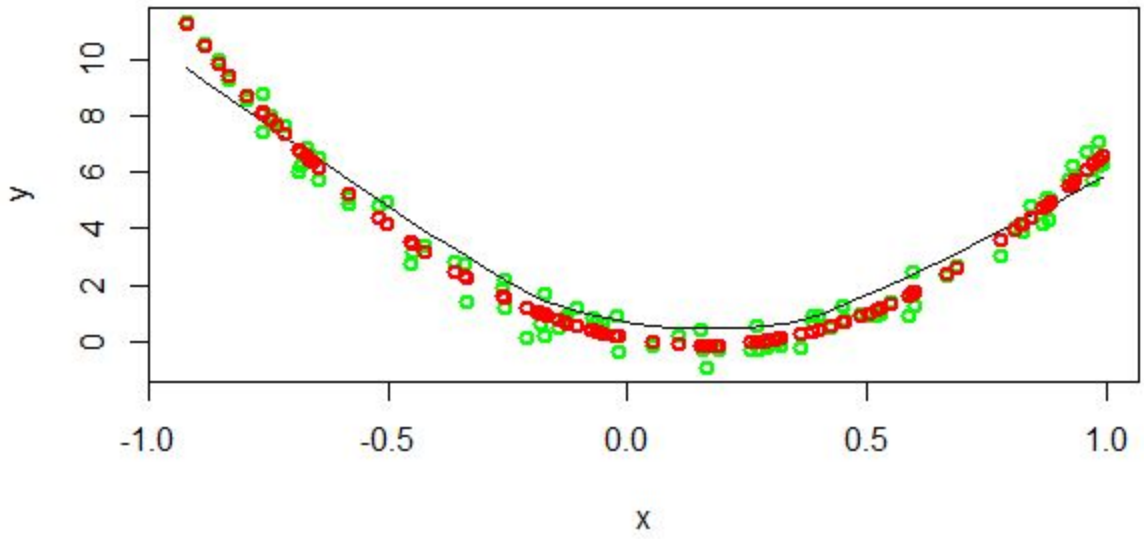


2. Fit a quadratic regression model to the data. Show the data scatter plot on the same figure with the values predicted by the quadratic model.

- **Code:-**

```
qm.fit = lm(y~poly(x,2), data=data1)
Qm.fit
z = predict(qm.fit, data=data1)
points(x, z, col="red", lwd=2)
```

- **Here, Green points - Actual data points & Red Points - Predicted values**



3.

a) Method 1 (Step function Learner)

- For the purpose of reproducing the result, I have used sed.seed(1). If you want to get a different result then don't use the set.seed() function.
- U = -0.4146425 ('U' is a Random Number uniformly on the interval spanned by min and max of inputs)

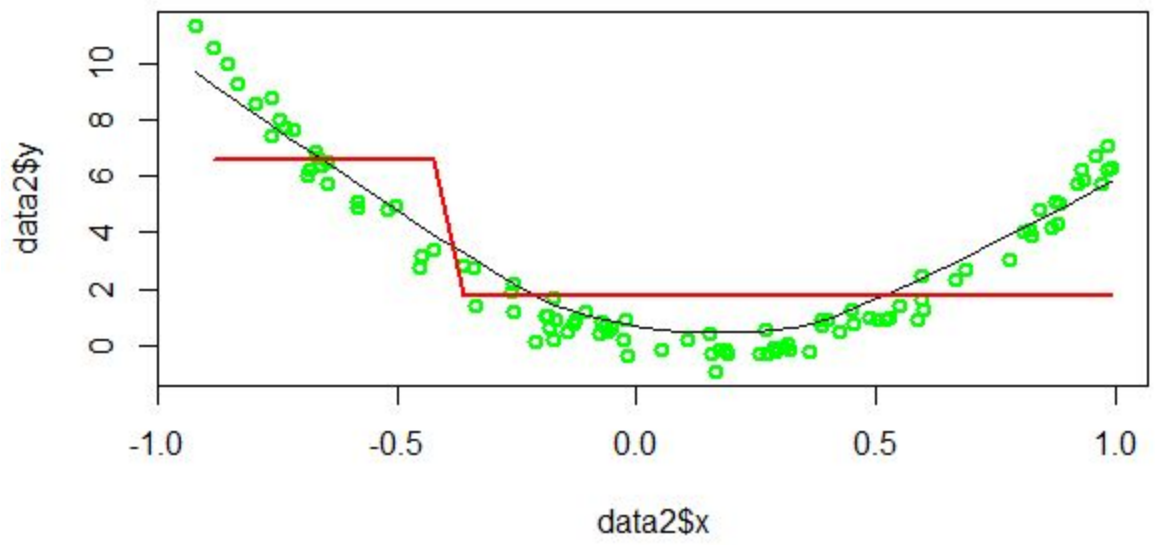
Inference:-

- As we can see from the figure, ‘f’ is not a strong learner, because the given data in the hand looks to be quadratic and we are trying to fit a step function which will not gonna fit the data completely. Hence it is not recommended to fit the step function in this case.

Code:-

```
#Sorting the data
set.seed(1)
data2 = data1[order(data1$x),]
#Selecting a random uniform number
U = runif(1, min(data1$x), max(data1$x))
#Selecting the cut points
v = c(min(data1$x), U, max(data1$x))
#Fitting the step function
step_fit = lm(data2$y ~ cut(data2$x,v))
#Predictions
preds = predict(step_fit, data2)
#Plots
scatter.smooth(data2$x,data2$y, col="green", lwd=2)
lines(data2$x, preds, col="red", lwd=2)
```

- **Here, Green points - Actual data points & Red Line - Step function learner**



b) Function (Step function learner)

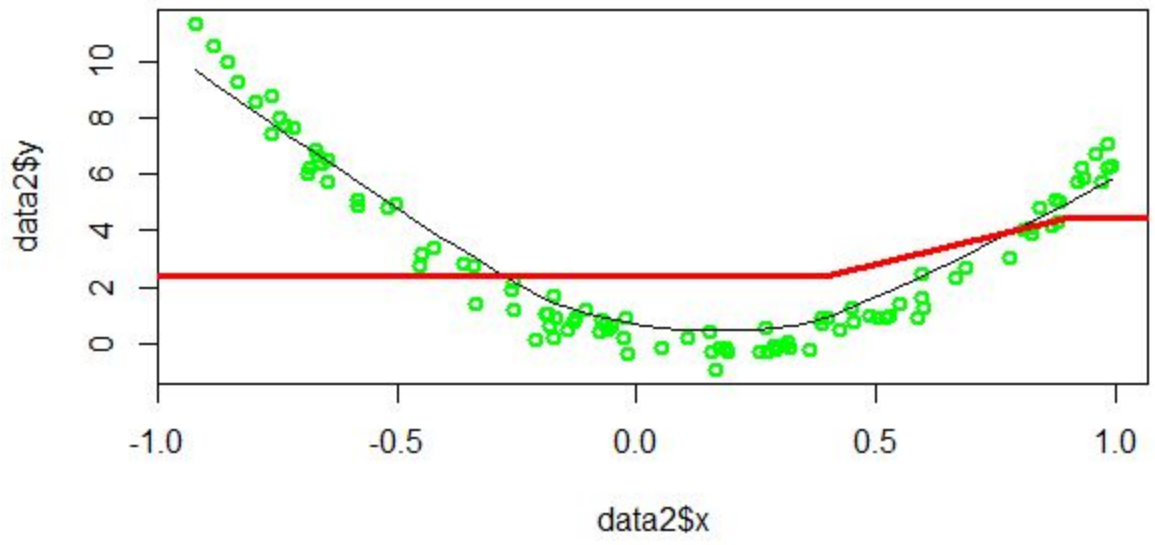
Code:-

```
step = function(input, data){
  #This function takes as input 2 arguments i.e. 'input' and 'data'.

  #Input Arguments
  # data - It is the data to which we are trying to fit the step function model.
  # input - It is either a scalar/vector of values for which the function will return the predicted
  #values.

  #Output
  #This function returns a scalar/vector of predicted values for the given inputs

  #Sorting the data
  data1 = data[order(data$x),]
  #Selecting 1 random uniform number
  U = runif(1, min(data$x), max(data$x))
  #Defining 3 cut points
  v = c(min(data$x), U, max(data$x))
  #Fitting the step function
  step_fit = lm(data1$y ~ cut(data1$x,v))
  preds = predict(step_fit, data1)
  unique_value = unique(preds)
  #Initialize the output
  output = matrix(0,1,length(input))
  for (i in 1:length(input)) {
    if(input[i]<=U){
      output[i] = unique_value[2]
    }
    if(input[i]>U){
      output[i] = unique_value[3]
    }

  }
  return(output)
}
```

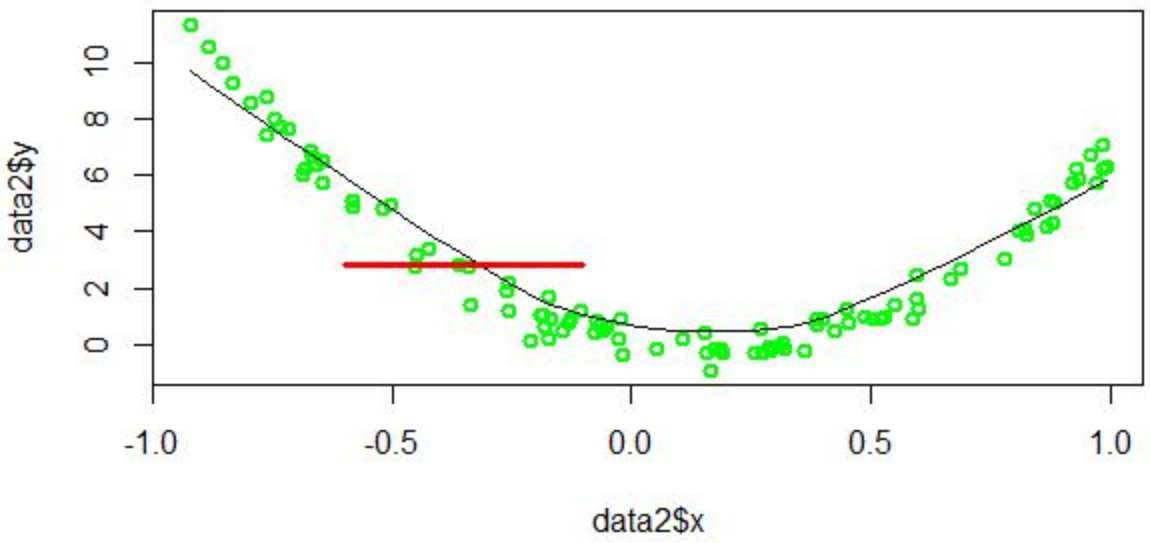
c) Code

```
input1 = c(-2,-3,-0.7,-0.1,0.4,0.9, 2,3)
input2 = c(-0.1,-0.2,-0.3,-0.4,-0.5,-0.6)
input3 = c(-0.98,-0.81,-0.5,-0.4,-0.3,0.3,0.4,0.5,0.81,0.98)
output1 = step(input1, data1)
output2 = step(input2, data1)
output3 = step(input3, data1)

scatter.smooth(data2$x,data2$y, col="green", lwd=2)
lines(input1,output1, col="red", lwd=3, type='l')
scatter.smooth(data2$x,data2$y, col="green", lwd=2)
lines(input2,output2, col="red", lwd=3, type='l')
scatter.smooth(data2$x,data2$y, col="green", lwd=2)
lines(input3,output3, col="red", lwd=3, type='l')
```

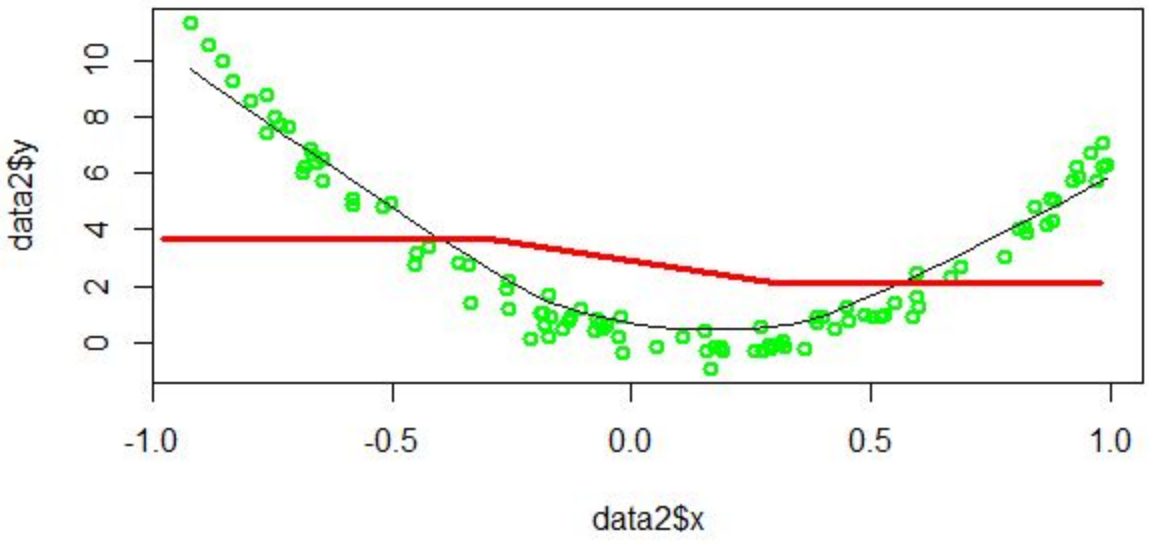
Plot 1



Plot 2


data2$y
0
2
4
6
8
10
-1.0
-0.5
0.0
0.5
1.0
data2$x

Plot 6


data2$y
0
2
4
6
8
10
-1.0
-0.5
0.0
0.5
1.0
data2$x