Programme	:	**B Tech – ECE and ECM**	Semester	:	**Win 2022**
Course	:	**Essentials of Data Analytics Lab**	Code	:	**CSE3506**
Faculty	:	**Gobinath N**	Slot	:	**L51 + L52**

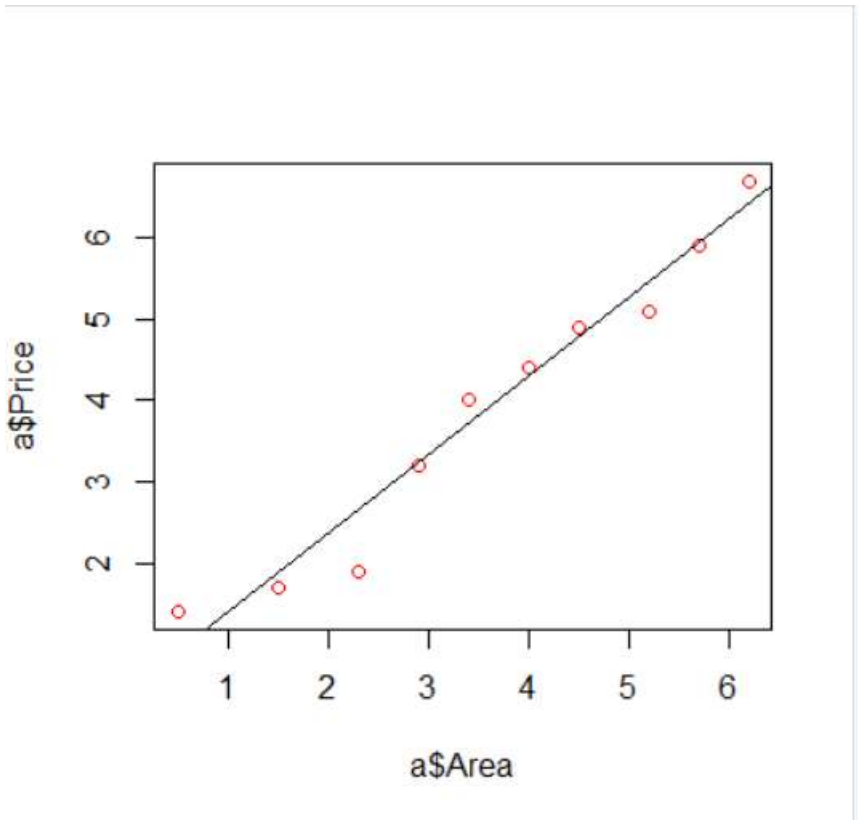
Ex_07_Gradient_descent

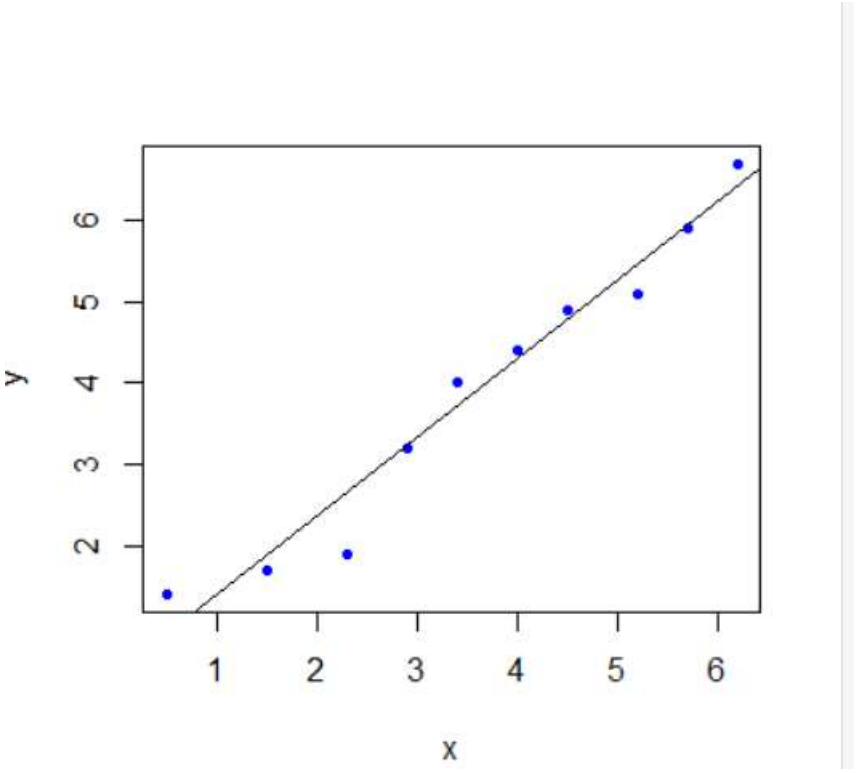
According to the Gaussian Noise concept, there are two kinds of errors, systematic and random. Simply stated, a systematic error is something that follows a certain direction. It is consistent in nature, and predictable.

On the other hand, a random error is noise in our distribution. Once we've taken care of the systematic noise component, the best predictor is obtained when the random noise is minimized. Putting it another way, the best predictor is the one with the tightest distribution (smallest variance) around the predicted value. Minimizing the least squared loss is the same thing as minimizing the variance! That explains why the least squared loss works for a wide range of problems. The underlying noise is very often Gaussian, because of the CLT (Central Limit Theorem), and minimizing the squared error turns out to be the right thing to do!

In this, we use gradient function to compare the learning rate(both Intercept and coefficient) with the linear regression model of the same

Plots:





```
> gradDesc(a$Area, a$Price, 0.001, 0.001, 32, 2500000)
[1] "Optimal intercept: 0.417015774501525 Optimal slope: 0.967675200413919"
>
>
>
>
>
>
>
>
>
>
> coefficients(mdl)
(Intercept)      a$Area
  0.4170158   0.9676752
```

Annex:

Code:

```
rm(list=ls())
setwd("C:\\Users\\Rituraj Anand\\Desktop\\Sem6\\CSE3506\\LAB\\Lab 7")
a=read.csv("Gradient_Descent.csv")
plot(a$Area,a$Price,col='red')
mdl<-lm(a$Price~a$Area,data=a)
coefficients(mdl)
pdct<-predict(mdl)
abline(mdl)
errors<-unname((a$Price-pdct)^2)
sum(errors)/length(a$Price)

#(x,y,learning rate, convergence criteria,no.of runs, maximum iterations)
gradDesc <- function(x, y, learn_rate, conv_threshold, n,max_iter) {
  plot(x, y, col = "blue", pch = 20)
  m <- runif(1, 0, 1)
  c <- runif(1, 0, 1)
  yhat <- m * x + c
  MSE <- sum((y - yhat) ^ 2) / n
  converged = F
```

```
 iterations = 0
 converged = F
 iterations = 0
 while(converged == F) {
  ## Implement the gradient descent algorithm
  m_new <- m - learn_rate * ((1 / n) * (sum((yhat - y) * x)))
  c_new <- c - learn_rate * ((1 / n) * (sum(yhat - y)))
  m <- m_new
  c <- c_new
  yhat <- m * x + c
  MSE_new <- sum((y - yhat) ^ 2) / n
  if(MSE - MSE_new <= conv_threshold) {
   abline(c, m)
   converged = T
   return(paste("Optimal intercept:", c, "Optimal slope:", m))
  }
  iterations = iterations + 1
  if(iterations > max_iter)
  {
   abline(c, m)
   converged = T
   return(paste("Optimal intercept:", c, "Optimal slope:", m))
  }
 }
}
gradDesc(a$Area, a$Price, 0.001, 0.001, 32, 2500000)
```

Result and Inference:

We saw, when the iteration increases, the error between contiguous solution decreases and whenever the error reaches < 0.0005, we accept the solution.