Assignment No 2

To perform Supervised Learning – Regression.

Aim
Generate a proper 2-D data set of N points. Split the data set into Training Data set and Test Data set & perform linear regression.

Objective(s)	
1	Perform linear regression analysis with Least Squares Method
2	Plot the graphs for Training MSE and Test MSE and comment on Curve Fitting and Generalization Error.
3	Verify the Effect of Data Set Size and Bias-Variance Tradeoff.
4	Apply Cross Validation and plot the graphs for errors. Apply Subset Selection Method and plot the graphs for errors.

Theory

In statistical modeling, regression analysis is a statistical process for estimating the relationships among variables. It includes many techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Most commonly, regression analysis estimates the conditional expectation of the dependent variable given the independent variables – that is, the average value of the dependent variable when the independent variables are fixed. Less commonly, the focus is on a quantile, or other location parameter of the conditional distribution of the dependent variable given the independent variables. In all cases, the estimation target is a function of the independent variables called the regression function. In regression analysis, it is also of interest to characterize the variation of the dependent variable around the regression function which can be described by a probability distribution**.**

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer causal relationships between the independent and dependent variables.

Many techniques for carrying out regression analysis have been developed. Familiar methods such as linear regression and ordinary least squares regression are parametric, in that the regression function is defined in terms of a finite number of unknown parameters that are estimated from the data. Nonparametric regression refers to techniques that allow the regression function to lie in a specified set of functions, which may be infinite-dimensional.

Linear Regression:

Linear regression is the most basic and commonly used predictive analysis. Regression estimates are used to describe data and to explain the relationship between one dependent variable and one or more independent variables.

At the center of the regression analysis is the task of fitting a single line through a scatter plot.

The simplest form with one dependent and one independent variable is defined by the formula

$$y = c + b*x,$$

where y = estimated dependent,
c = constant, b = regression coefficients,
x = independent variable.

However linear regression analysis consists of more than just fitting a linear line through a cloud of data points. It consists of 3 stages –

(1) analyzing the correlation and directionality of the data,
(2) estimating the model, i.e., fitting the line, and
(3)evaluating the validity and usefulness of the model.

Sometimes the dependent variable is also called a criterion variable, endogenous variable, prognostic variable, or regressand. The independent variables are also called exogenous variables, predictor variables or regressors.

There are 3 major uses for regression analysis –
(1) causal analysis,
(2) forecasting an effect,
(3) trend forecasting.

Other than correlation analysis, which focuses on the strength of the relationship between two or more variables, regression analysis assumes a dependence or causal relationship between one or more independent and one dependent variable.

Firstly, it might be used to identify the strength of the effect that the independent variable(s) have on a dependent variable. Typical questions are what is the strength of relationship between dose and effect, sales and marketing spend, age and income.

Secondly, it can be used to forecast effects or impacts of changes. That is regression analysis helps us to understand how much will the dependent variable change, when we change one or more independent variables. Typical questions are how much additional Y do I get for one additional unit X.

Thirdly, regression analysis predicts trends and future values. The regression analysis can be used to get point estimates. Typical questions are what will the price for gold be in 6 month from now? What is the total effort for a task X?

Many Names of Linear Regression

When you start looking into linear regression, things can get very confusing.The reason is because linear regression has been around for so long (more than 200 years). It has been studied from every possible angle and often each angle has a new and different name.

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression.

Different techniques can be used to prepare or train the linear regression equation from data, the most common of which is called Ordinary Least Squares. It is common to therefore refer to a model prepared this way as Ordinary Least Squares Linear Regression or just Least Squares Regression.

Now that we know some names used to describe linear regression, let's take a closer look at the representation used.

Simple Linear Regression

When we have a single input attribute (x) and we want to use linear regression, this is called simple linear regression.

If we had multiple input attributes (e.g. x1, x2, x3, etc.) This would be called multiple linear regression. The procedure for linear regression is different and simpler than that for multiple linear regression.

In this section we are going to create a simple linear regression model from our training data, then make predictions for our training data to get an idea of how well the model learned the relationship in the data.

With simple linear regression we want to model our data as follows:

$$y = B0 + B1 * x$$

This is a line where y is the output variable we want to predict, x is the input variable we know and B0 and B1 are coefficients that we need to estimate that move the line around.

Technically, B0 is called the intercept because it determines where the line intercepts the y-axis. In machine learning we can call this the bias, because it is added to offset all predictions that we make. The B1 term is called the slope because it defines the slope of the line or how x translates into a y value before we add our bias.

The goal is to find the best estimates for the coefficients to minimize the errors in predicting y from x.

Simple regression is great, because rather than having to search for values by trial and error or calculate them analytically using more advanced linear algebra, we can estimate them directly from our data.

We can start off by estimating the value for B1 as:

$$B1 = \frac{\sum (Xi - \bar{X}) * (Yi - \bar{Y})}{\sum (Xi - \bar{X})^2}$$

Where mean() is the average value for the variable in our dataset. The xi and yi refer to the fact that we need to repeat these calculations across all values in our dataset and i refers to the i'th value of x or y.

We can calculate B0 using B1 and some statistics from our dataset, as follows:

$$B0 = \bar{Y} - (B1 * \bar{X})$$

Estimating Slope (B1)

Let's start with the top part of the equation, the numerator. First we need to calculate the mean value of x and y. The mean is calculated as: sum(x) / n Where n is the number of values (5 in this case). Let's calculate the mean value of our x and y variables:

x =3 $\bar{Y}$= 2.8

We now have the parts for calculating the numerator. All we need to do is multiple the error for each x with the error for each y and calculate the sum of these multiplications

	x - mean(x)	y - mean(y)	Multiplication
2	-2	-1.8	3.6
3	-1	0.2	-0.2
4	1	0.2	0.2
5	0	-0.8	0
6	2	2.2	4.4

.

Summing the final column we have calculated **our numerator as 8.**

Now we need to calculate the bottom part of the equation for calculating B1, or the denominator. This is calculated as the sum of the squared differences of each x value from the mean.

We have already calculated the difference of each x value from the mean, all we need to do is square each value and calculate the sum.

Calculating the sum of these squared values gives us up **denominator of 10**

Now we can calculate the value of our slope.

B1 = 8 / 10 so further B1 = 0.8

Estimating Intercept (B0)

This is much easier as we already know the values of all of the terms involved.

$$B0 = \bar{Y} - (B1 * \bar{X})$$

or

B0 = 2.8 – 0.8 * 3 , or further B0 = 0.4

Making Predictions

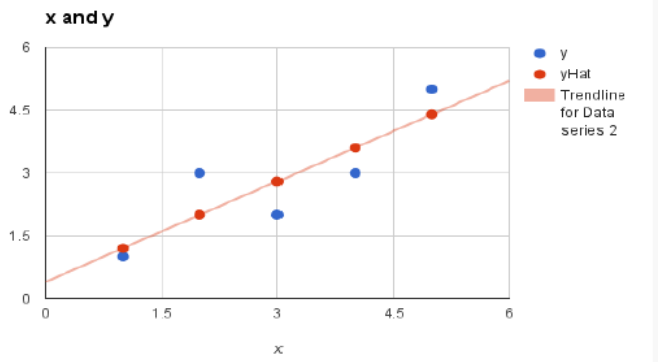
We now have the coefficients for our simple linear regression equation.

y = B0 + B1 * x or
y = 0.4 + 0.8*x

Let's try out the model by making predictions for our training data.

	x	y	predicted y
1			
2	1	1	1.2
3	2	3	2
4	4	3	3.6
5	3	2	2.8
6	5	5	4.4

We can plot these predictions as a line with our data. This gives us a visual idea of how well the line models our data.



Simple Linear Regression Model

Estimating Error

We can calculate a error for our predictions called the Root Mean Squared Error or RMSE.

Where sqrt() is the square root function, p is the predicted value and y is the actual value, i is the index for a specific instance, n is the number of predictions, because we must calculate the error across all predicted values

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

First we must calculate the difference between each model prediction and the actual y values. We can easily calculate the square of each of these error values (error*error or error^2).

	pred-y	y	error	Squared error
1				
2	1.2	1	0.2	0.04
3	2	3	-1	1
4	3.6	3	0.6	0.36
5	2.8	2	0.8	0.64
6	4.4	5	-0.6	0.36

The sum of these errors is 2.4 units, dividing by n and taking the square root gives us:
RMSE = 0.692 Or, each prediction is on average wrong by about 0.692 units.

Least square method

The basic idea of the method of least squares is easy to understand. It may seem unusual that when several people measure the same quantity, they usually do not obtain the same results. In fact, if the same person measures the same quantity several times, the results will vary. What then is the best estimate for the true measurement? The method of least squares gives a way to find the best estimate, assuming that the errors (i.e. the differences from the true value) are random and unbiased.

In statistics and mathematics, linear least squares is an approach fitting a mathematical or statistical model to data in cases where the idealized value provided by the model for any data point is expressed linearly in terms of the unknown parameters of the model. The resulting fitted model can be used to summarize the data, to predict unobserved values from the same system, and to understand the mechanisms that may underlie the system.

Mathematically, linear least squares is the problem of approximately solving an overdetermined system of linear equations, where the best approximation is defined as that which minimizes the sum of squared differences between the data values and their corresponding modeled values. The approach is called linear least squares since the assumed function is linear in the parameters to be estimated. Linear least squares problems are convex and have a closed-form solution that is unique, provided that the number of data points used for fitting equals or exceeds the number of unknown parameters, except in special degenerate situations. In contrast, non-linear least squares problems generally must be solved by an iterative procedure, and the problems can be non-convex with multiple optima for the objective function. If prior distributions are available, then even an underdetermined system can be solved using the Bayesian MMSE estimator

Conclusion
Linear Regression has been performed successfully.

Sample Program for reference

```
rm(list=ls())

input <- read.csv("F:/Rajasree/R/2019_20SEMI/ML/Regression/DataSets/sample.csv")
attach(input)
names(input)
head(input,3)
plot(runs~at_bats,main="Runs Vs At_bats",xlim=c(1,6),ylim=c(1,6))
cor(runs,at_bats)
plotSS <- function(x, y, showSquares = FALSE, leastSquares = FALSE)
{
 plot(y~x, asp = 1, xlab=paste(substitute(x)), ylab=paste(substitute(y)))
 if(leastSquares)
 {
  y.hat <- m1$fit
 }
 else {
  cat("Click any two points to make a line.\n")
  pt1 <- locator(1)
  points(pt1$x, pt1$y, pch = 8, col = "red")
  pt2 <- locator(1)
  points(pt2$x, pt2$y, pch = 8, col = "red")
  pts <- data.frame("x" = c(pt1$x, pt2$x),"y" = c(pt1$y, pt2$y))
```

```
    m1 <- lm(y ~ x, data = pts)
    y.hat <- predict(m1, newdata = data.frame(x))
    # title(paste("b0 = ", pt1$y-(pt2$y-pt1$y)/(pt2$x-pt1$x)*pt1$x, ", b1 = ", (pt2$y-
pt1$y)/(pt2$x-pt1$x)))
  }
  r <- y - y.hat
  abline(m1)
  oSide <- x - r
  LLim <- par()$usr[1]
  RLim <- par()$usr[2]
  oSide[oSide < LLim | oSide > RLim] <- c(x + r)[oSide < LLim | oSide > RLim]
  n <- length(y.hat)
  for(i in 1:n)
  {
    lines(rep(x[i], 2), c(y[i], y.hat[i]), lty = 2, col = "blue")
    if(showSquares)
    {
      lines(rep(oSide[i], 2), c(y[i], y.hat[i]), lty = 3, col = "orange")
      lines(c(oSide[i], x[i]), rep(y.hat[i],2), lty = 3, col = "orange")
      lines(c(oSide[i], x[i]), rep(y[i],2), lty = 3, col = "orange")
    }
  }
  SS <- round(sum(r^2), 3)
  cat("\r ")
  print(m1)
  cat("Sum of Squares: ", SS, "\n")
}
plotSS(x = at_bats, y = runs)
plotSS(x = at_bats, y = runs, showSquares = TRUE)
fit1 <- lm(runs ~ at_bats)
summary(fit1)
plotSS(x = at_bats, y = runs, leastSquares = TRUE)
```

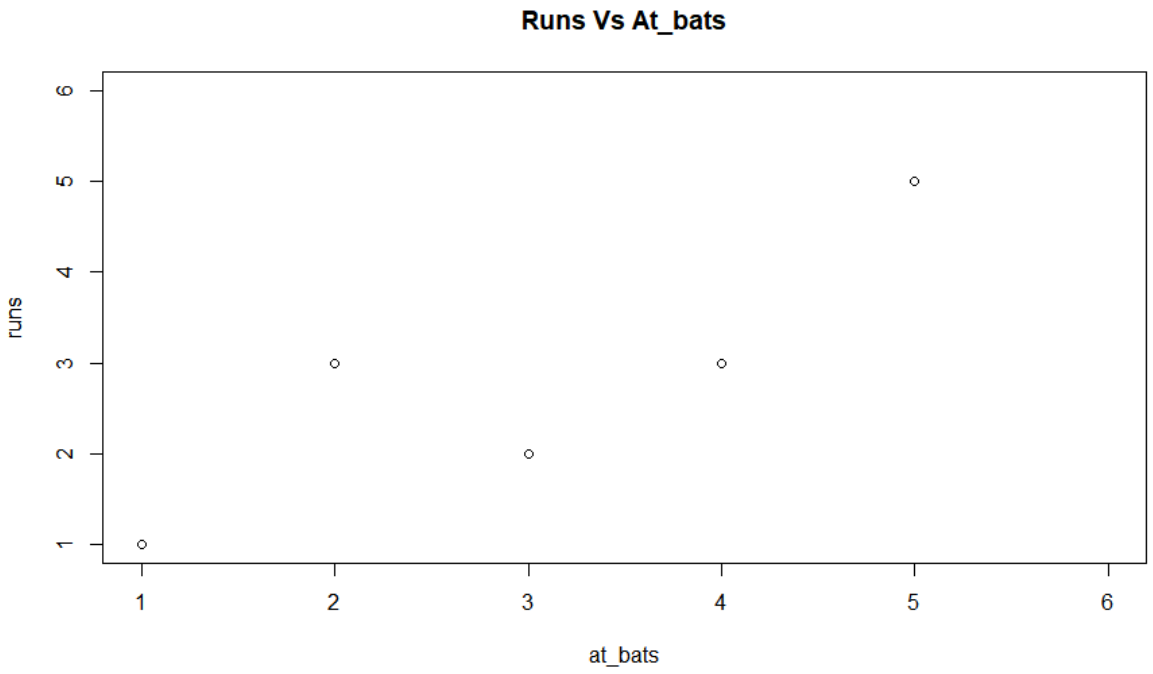
```
rm(list=ls())
input <- read.csv("C:/Users/aditya/Downloads/sample.csv")
attach(input)
names(input)
```

```
[1] "team"         "runs"         "at_bats"      "hits"         "homeruns"     "bat_avg"      "strikeouts"
 [8] "stolen_bases" "wins"         "new_onbase"   "new_slug"     "new_obs"
```

```
head(input,3)
```

```
    team runs at_bats hits homeruns bat_avg strikeouts stolen_bases wins new_onbase new_slug
new_obs
1  Texas Rangers    1       1   15      210   0.283        930           143   96       0.340
0.460   0.800
2 Boston Red Sox    3       2   16      203   0.280       1108           102   90       0.349
0.461   0.810
3 Detroit Tigers    3       4   15      169   0.277       1143            49   95       0.340
0.434   0.773
```

```
plot(runs~at_bats,main="Runs Vs At_bats",xlim=c(1,6),ylim=c(1,6))
```




```
cor(runs,at_bats)
```
0.8528029

```
plotSS <- function(x, y, showSquares = FALSE, leastSquares = FALSE)
+ {
+     plot(y~x, asp = 1, xlab=paste(substitute(x)), ylab=paste(substitute(
y)))
+     if(leastSquares)
+     {
+         y.hat <- m1$fit
+     }
+     else {
+         cat("Click any two points to make a line.\n")
+         pt1 <- locator(1)
+         points(pt1$x, pt1$y, pch = 8, col = "red")
+         pt2 <- locator(1)
+         points(pt2$x, pt2$y, pch = 8, col = "red")
+         pts <- data.frame("x" = c(pt1$x, pt2$x),"y" = c(pt1$y, pt2$y))
+         m1 <- lm(y ~ x, data = pts)
+         y.hat <- predict(m1, newdata = data.frame(x))
+         # title(paste("b0 = ", pt1$y-(pt2$y-pt1$y)/(pt2$x-pt1$x)*pt1$x,
", b1 = ", (pt2$y-pt1$y)/(pt2$x-pt1$x)))
+     }
+     r <- y - y.hat
+     abline(m1)
+     oSide <- x - r
+     LLim <- par()$usr[1]
+     RLim <- par()$usr[2]
+     oSide[oSide < LLim | oSide > RLim] <- c(x + r)[oSide < LLim | oSide
> RLim]
+     n <- length(y.hat)
+     for(i in 1:n)
+     {
+         lines(rep(x[i], 2), c(y[i], y.hat[i]), lty = 2, col = "blue")
+         if(showSquares)
+         {
+             lines(rep(oSide[i], 2), c(y[i], y.hat[i]), lty = 3, col = "o
range")
+             lines(c(oSide[i], x[i]), rep(y.hat[i],2), lty = 3, col = "or
ange")
+             lines(c(oSide[i], x[i]), rep(y[i],2), lty = 3, col = "orange
")
+         }
+     }
+     SS <- round(sum(r^2), 3)
+     cat("\r ")
+     print(m1)
+     cat("Sum of Squares: ", SS, "\n")
+ }
```

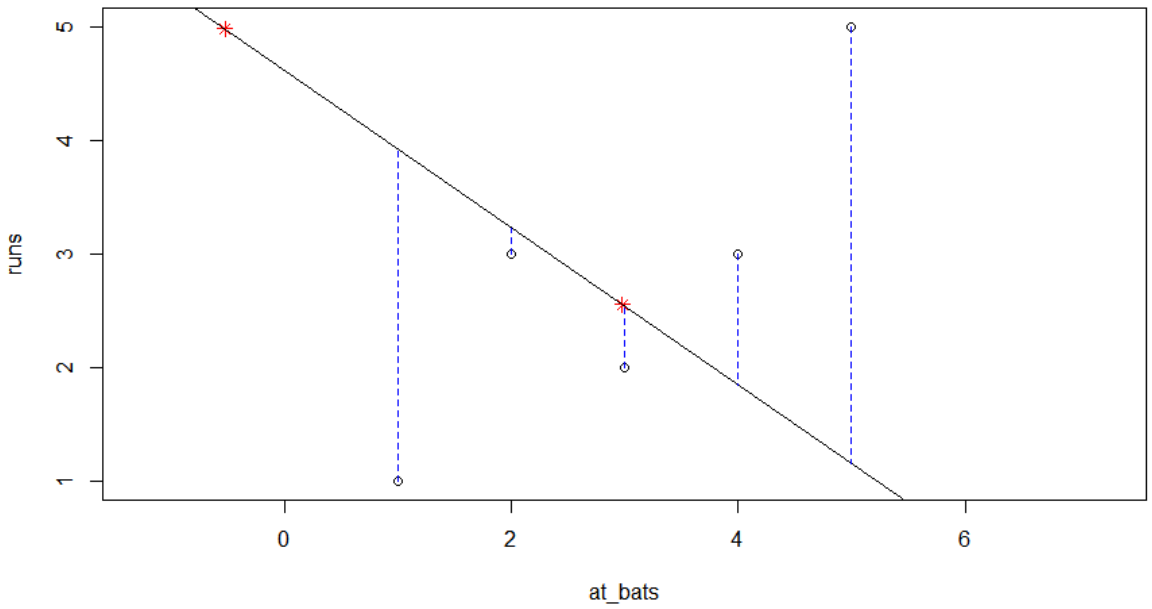
```
plotSS(x = at_bats, y = runs)

Click any two points to make a line.

Call:
lm(formula = y ~ x, data = pts)

Coefficients:
(Intercept)            x
     4.6136      -0.6915

Sum of Squares:  24.987
```

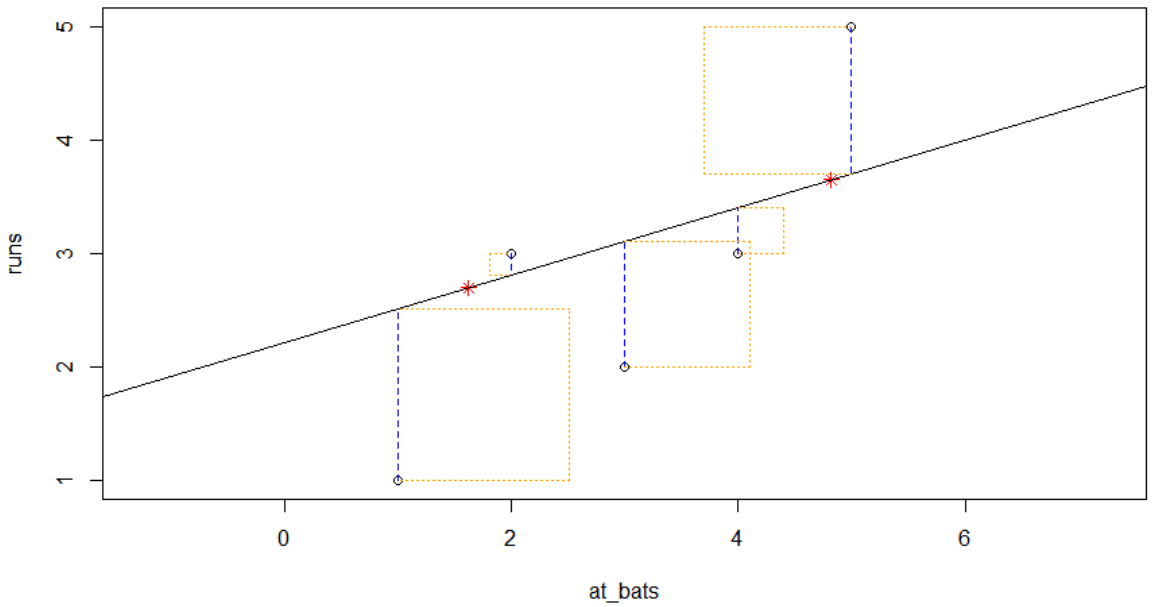



```
plotSS(x = at_bats, y = runs, showSquares = TRUE)
Click any two points to make a line.

Call:
lm(formula = y ~ x, data = pts)

Coefficients:
(Intercept)            x
     2.2125       0.2974

Sum of Squares:  5.39
```




```
fit1 <- lm(runs ~ at_bats)
summary(fit1)
```

```
Call:
lm(formula = runs ~ at_bats)

Residuals:
   1    2    3    4    5
-0.2  1.0 -0.6 -0.8  0.6

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.4000     0.9381   0.426   0.6986
at_bats       0.8000     0.2828   2.828   0.0663 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.8944 on 3 degrees of freedom
Multiple R-squared:  0.7273,   Adjusted R-squared:  0.6364
F-statistic:     8 on 1 and 3 DF,  p-value: 0.06628
```