SURESH KUMAR PRAJAPATI DATE-11/03/2024

ENROLLMENT NO. -CUSB23022220008
DEPARTMENT OF STATISTICS
(MASTER IN DATA SCIENCE AND APPLIED STATISTICS)

Environmental Sciences :-

In a 1976 study exploring the relationship between water quality and land use, Haith (1976) obtained the measurements on 20 river basins in New York State. A question of interest here is how the land use around a river basin contributes to the water pollution as measured by the mean nitrogen concentration (mglliter).

Ildustrial Production :-

Nambe Mills in Santa Fe, New Mexico, makes a line of tableware made from sand casting a special alloy of metals. After casting, the pieces go through a series of shaping, grinding, buffing, and polishing steps. Data was collected for 59 items produced by the company. The relation between the polishing time and the product diameters and the product types (Bowl, Casserole, Dish, Tray, and Plate) are used to estimate the polishing time for new products which are designed or suggested for design and manufacturs. The variables representing product types are coded as binary variables (1 corresponds to the type and 0 otherwise). Diam is the diameter of the item (in inches), polishing time is measured in minutes, and price in dollars. The polishing time is the major item in the cost of the product. The production decision will be based on the estimated time of polishing. The data is obtained from the DASL library.

VARIABLE	DEFINITION
Y	Mean nitrogen concentration (mglliter) based on samples taken at regular intervals during the spring, summer, and fall months.
X1	Agriculture: percentage of land area currently in agricultural use
X2	Forest: percentage of forest land
X3	Residential: percentage of land area in residential use.
X4	CommercialIIndustrial: percentage of land area in either commercial or industrial use

```
> rm(list = ls())
> setwd("C:\\Users\\Admin\\OneDrive\\Desktop\\river")
> data=read.csv("river_data.csv")
> data
    y x1 x2  x3  x4
1 1.10 26 63 1.2 0.29
```

```
2  1.01 29 57  0.7 0.09
3  1.90 54 26  1.8 0.58
4  1.00  2 84  1.9 1.98
5  1.99  3 27 29.4 3.11
6  1.42 19 61  3.4 0.56
7  2.04 16 60  5.6 1.11
8  1.65 40 43  1.3 0.24
9  1.01 28 62  1.1 0.15
10 1.21 26 60  0.9 0.23
11 1.33 26 53  0.9 0.18
12 0.75 15 75  0.7 0.16
13 0.73  6 84  0.5 0.12
14 0.80  3 81  0.8 0.35
15 0.76  2 89  0.7 0.35
16 0.87  6 82  0.5 0.15
17 0.80 22 70  0.9 0.22
18 0.87  4 75  0.4 0.18
19 0.66 21 56  0.5 0.13
20 1.25 40 49  1.1 0.13
> modl=lm(y~x1+x2+x3+x4,data)
> modl

Call:
lm(formula = y ~ x1 + x2 + x3 + x4, data = data)

Coefficients:
(Intercept)        x1        x2        x3
  1.722214    0.005809   -0.012968   -0.007227
        x4
  0.305028

> mdls=summary(modl)
> mdls

Call:
lm(formula = y ~ x1 + x2 + x3 + x4, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.49404 -0.13180  0.01951  0.08287  0.70480

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.722214   1.234082   1.396   0.1832
x1           0.005809   0.015034   0.386   0.7046
x2          -0.012968   0.013931  -0.931   0.3667
x3          -0.007227   0.033830  -0.214   0.8337
x4           0.305028   0.163817   1.862   0.0823 .
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2649 on 15 degrees of freedom
```

Multiple R-squared: 0.7094, Adjusted R-squared: 0.6319
F-statistic: 9.154 on 4 and 15 DF, p-value: 0.0005963

#Regression equation in matrix 1 as intercept I.e.null model there is no predictors.

```
> x=model.matrix(~x1+x2+x3+x4,data)
> x
   (Intercept) x1 x2   x3   x4
1            1 26 63  1.2 0.29
2            1 29 57  0.7 0.09
3            1 54 26  1.8 0.58
4            1  2 84  1.9 1.98
5            1  3 27 29.4 3.11
6            1 19 61  3.4 0.56
7            1 16 60  5.6 1.11
8            1 40 43  1.3 0.24
9            1 28 62  1.1 0.15
10           1 26 60  0.9 0.23
11           1 26 53  0.9 0.18
12           1 15 75  0.7 0.16
13           1  6 84  0.5 0.12
14           1  3 81  0.8 0.35
15           1  2 89  0.7 0.35
16           1  6 82  0.5 0.15
17           1 22 70  0.9 0.22
18           1  4 75  0.4 0.18
19           1 21 56  0.5 0.13
20           1 40 49  1.1 0.13
attr(,"assign")
[1] 0 1 2 3 4
#Stract the y from the data
> y=data$y
> y
 [1] 1.10 1.01 1.90 1.00 1.99 1.42 2.04 1.65 1.01 1.21
[11] 1.33 0.75 0.73 0.80 0.76 0.87 0.80 0.87 0.66 1.25
```

To construct:-

$(X^TX)^{-1}$. t ( ) does transpose and %*% does matrix multiplication. solve (A) computes (A^{-1}) while solve (A, b) solves Ax=b

```
> xtx1=solve(t(x)%*%x)
> xtx1
            (Intercept)          x1           x2
(Intercept) 21.70014032 -0.257809328 -0.243655844
x1          -0.25780933  0.003220489  0.002851994
x2          -0.24365584  0.002851994  0.002765471
x3          -0.48497199  0.005637123  0.005598278
x4          -0.03524849  0.001511668 -0.001322999

                    x3           x4
(Intercept) -0.484971990 -0.035248486
x1           0.005637123  0.001511668
x2           0.005598278 -0.001322999
x3           0.016307226 -0.039794590
x4          -0.039794590  0.382376260
```

$\Sigma \epsilon_i^2 = (Y-X\beta)^T(Y-X\beta)$

Differentiating w.r.t. to Beta and setting zero we find Beta cap

$X^TX\hat{\beta} = X^TY$ it is normal equation

$\hat{\beta} = (X^TX)^{-1}X^TY$

Get B cap, using $(X^T X)^{-1} X^T y$:

```
> B=xtx1%*%t(x)%*%y
> B
                [,1]
(Intercept) 1.722213529
x1          0.005809126
x2         -0.012967887
x3         -0.007226768
x4          0.305027765
```

The above method is very bad to compute Beta cap. Suppose for large data set then easily we find to find same result.

```
> t=solve(crossprod(x,x),crossprod(x,y))
> t
                [,1]
(Intercept) 1.722213529
x1          0.005809126
x2         -0.012967887
x3         -0.007226768
x4          0.305027765
```

Below these term also find.

```
> names(modl)
 [1] "coefficients"  "residuals"     "effects"
 [4] "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "xlevels"
[10] "call"          "terms"         "model"
> mdls=summary(modl)
> names(mdls)
 [1] "call"          "terms"         "residuals"
 [4] "coefficients"  "aliased"       "sigma"
 [7] "df"            "r.squared"     "adj.r.squared"
[10] "fstatistic"    "cov.unscaled"
```

To Determine the σ from the summary,

$\sigma^2 = £(\hat{\epsilon}^T\hat{\epsilon})/(n-p) = R_{ss}/n-p$;Where n-p is degree of freedom.R_{ss} is resudial sum of square.

$\hat{\epsilon}^T\hat{\epsilon} = R_{SS} = Y^T(1-H)^T(1-H)Y = Y^T(1-H)Y$;H is a Hat matrix.

$H = X(X^TX)^{-1}X^T$; $\hat{Y} = HY$; $\hat{Y}$=Predicted value or fitted value.

```
> sqrt(deviance(modl)/df.residual(modl))
[1] 0.2649187
> mdls$sigma
[1] 0.2649187
```

To determine the extract $(X^TX)^{-1}$ and use it to compute the standard errors for the coefficents.

(diag() return the diagonal of a matrix):

$\hat{\beta}_{i-1} = \sqrt{(X^TX)_{ii}^{-1}}\hat{\sigma}$

```
> xtxi=mdls$cov.unscaled
> xtxi
            (Intercept)          x1          x2
(Intercept) 21.70014032 -0.257809328 -0.24365584
x1          -0.25780933  0.003220489  0.00285199
x2          -0.24365584  0.002851994  0.00276547
```

```
x3       -0.48497199          0.005637123      0.005598278
x4       -0.03524849          0.001511668     -0.001322999
              x3                x4
(Intercept) -0.484971990         -0.035248486
x1           0.005637123          0.001511668
x2           0.005598278         -0.001322999
x3           0.016307226         -0.039794590
x4          -0.039794590          0.382376260
```

To determine the summary of the data

```
> coeffi=sqrt(diag(xtx1))*0.2649187
> coeffi
(Intercept)        x1        x2        x3         x4
 1.23408163  0.01503396  0.01393148    0.03383005   0.16381666

> mdls$coefficients[,2]
(Intercept)        x1        x2        x3         x4
 1.23408166  0.01503396  0.01393148  0.03383005        0.16381667
```

To compute the R^2 :-

```
> 1-deviance(modl) /sum((y-mean(y))^2)
[1] 0.7093976
> mdls$r.squared
[1] 0.7093976
```

We create a new variable for the river data

```
> adiff=data$x1 -data$x2
> adiff
 [1] -37 -28  28 -82 -24 -42 -44  -3 -34 -34 -27 -60
[13] -78 -78 -87 -76 -48 -71 -35  -9
```

#Add the modal

```
> g=lm(y~x1+x2+x3+x4+adiff,data)
> g

Call:
lm(formula = y ~ x1 + x2 + x3 + x4 + adiff, data = data)

Coefficients:
(Intercept)        x1        x2        x3
  1.722214    0.005809   -0.012968   -0.007227
       x4       adiff
  0.305028         NA
> Adiffe=data$adiff+0.001*(runif(30)-0.5)
> Adiffe
```

```
numeric(0)
> g=lm(y~x1+x2+x3+x4+adiff,data)
> summary(g)

Call:
lm(formula = y ~ x1 + x2 + x3 + x4 + adiff, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.49404 -0.13180  0.01951  0.08287  0.70480

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.722214   1.234082   1.396   0.1832
x1           0.005809   0.015034   0.386   0.7046
x2          -0.012968   0.013931  -0.931   0.3667
x3          -0.007227   0.033830  -0.214   0.8337
x4           0.305028   0.163817   1.862   0.0823 .
adiff             NA         NA      NA       NA
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2649 on 15 degrees of freedom
Multiple R-squared:  0.7094,    Adjusted R-squared:  0.6319
F-statistic: 9.154 on 4 and 15 DF,  p-value: 0.0005963

> plot(data

> plot(modl)
Hit <Return> to see next plot: return()
Hit <Return> to see next plot: plot(g)
Hit <Return> to see next plot: return()
Hit <Return> to see next plot:
```