Untitled

ABDUL RAUF

2023-01-14

Contents

* LAB 8-11 *

```
__NAME__:-**ABDUL RAUF** \
__ENRL NO__:-**GL6092** \
__ROLL NO__:-**22DSMSA116** \
```

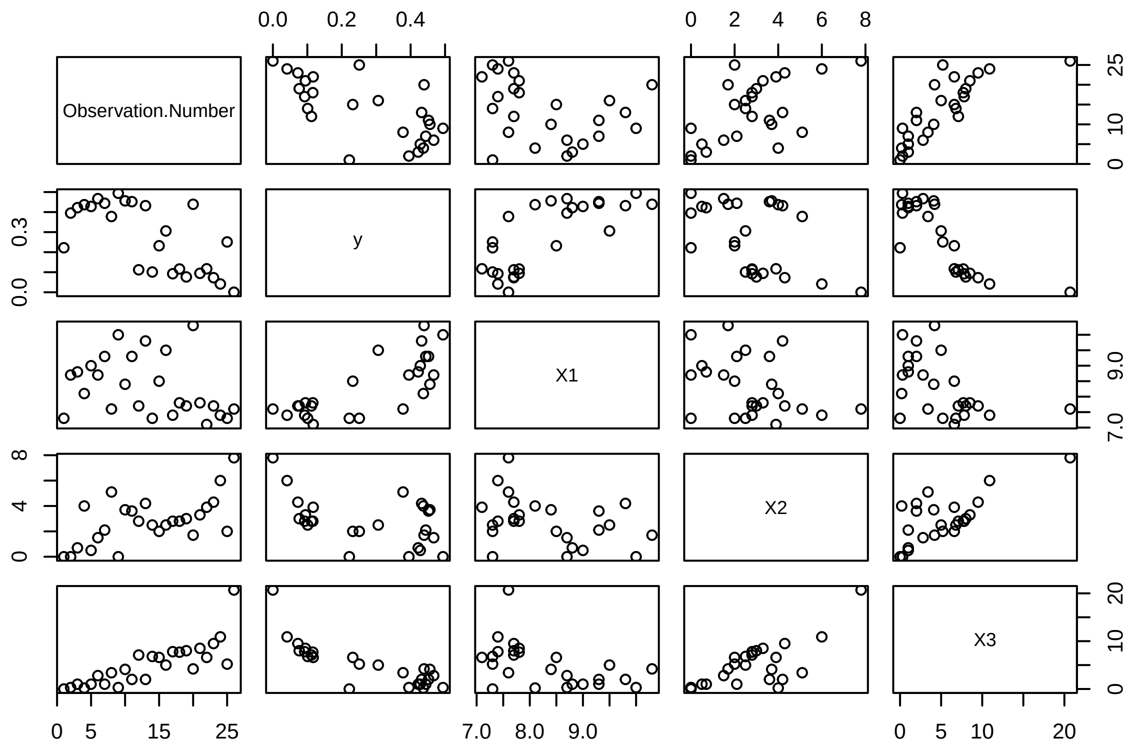
1 Validation set approach

1.1 reading the data from excel

```
solubility<-read.csv("solubility.csv")
head(solubility)
```

```
##   Observation.Number     y  X1  X2  X3
## 1                  1 0.222 7.3 0.0 0.0
## 2                  2 0.395 8.7 0.0 0.3
## 3                  3 0.422 8.8 0.7 1.0
## 4                  4 0.437 8.1 4.0 0.2
## 5                  5 0.428 9.0 0.5 1.0
## 6                  6 0.467 8.7 1.5 2.8
```

```
y=solubility$y
x1=solubility$X1
x2=solubility$x2
x3=solubility$X3
plot(solubility)
```



1.2 Dividing the data into train set and fitting it.

```
train=sample(26,13)
lm.fit<-lm(y~.,data = solubility, subset = train)

attach(solubility)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     y
```

```
## The following objects are masked from solubility (pos = 3):
##
##     Observation.Number, X1, X2, X3, y
```

```
## The following objects are masked from solubility (pos = 4):
##
##     Observation.Number, X1, X2, X3, y
```

```
## The following objects are masked from solubility (pos = 5):
##
##     Observation.Number, X1, X2, X3, y
```

```
## The following objects are masked from solubility (pos = 6):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 7):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 8):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 9):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 10):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 11):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 12):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 13):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 15):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 17):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 18):
##
##     Observation.Number, X1, X2, X3, y
```

```
mselm=mean((y-predict(lm.fit,solubility))[-train]^2) #MSE


lm.fit21<-lm(solubility$y~poly(solubility$X1,2)+poly(solubility$X2,2)+poly(solubility$X3,2),data = solu
lm.fit21
```

```
##
## Call:
## lm(formula = solubility$y ~ poly(solubility$X1, 2) + poly(solubility$X2,
##     2) + poly(solubility$X3, 2), data = solubility, subset = train)
##
## Coefficients:
##            (Intercept)  poly(solubility$X1, 2)1
##                0.20979                  0.46763
## poly(solubility$X1, 2)2  poly(solubility$X2, 2)1
##               -0.05219                  0.45812
## poly(solubility$X2, 2)2  poly(solubility$X3, 2)1
##                0.83352                 -1.27910
## poly(solubility$X3, 2)2
##               -1.14443
```

```r
mse2=mean((y-predict(lm.fit21,solubility))[-train]^2) #MSE of quadratic regression of all variables
mse2
```

```
## [1] 0.05536278
```

```r
lm.fit22<-lm(y~poly(X2,2),data = solubility,subset = train)
lm.fit22
```

```
##
## Call:
## lm(formula = y ~ poly(X2, 2), data = solubility, subset = train)
##
## Coefficients:
##  (Intercept)  poly(X2, 2)1  poly(X2, 2)2
##       0.2721        0.1844        0.5687
```

```r
mse22=mean((y-predict(lm.fit22,solubility))[-train]^2)  #MSE of quadratic regression of x2 variable

lm.fit23<-lm(y~poly(X3,2),data = solubility,subset = train)
lm.fit23
```

```
##
## Call:
## lm(formula = y ~ poly(X3, 2), data = solubility, subset = train)
##
## Coefficients:
##  (Intercept)  poly(X3, 2)1  poly(X3, 2)2
##       0.1682       -2.1707       -1.3666
```

```r
mse23=mean((y-predict(lm.fit23,solubility))[-train]^2)   #MSE of quadratic regression of x3 variable

mse=cbind(mselm,mse21,mse22,mse23)
mse
```

```
##            mselm     mse21      mse22     mse23
## [1,] 0.006742168 0.1732388 0.07934168 0.3992314
```

- MSE of the model having the no any higher degree polynomial is lowest among all of them.

1.3 fitting the data using poly() and calculating the MSE

```
#for all variables
n=10
mse_all=numeric(n)
for (i in 1:10) {
  M1=lm(solubility$y~poly(solubility$X1,i)+poly(solubility$X2,i)+poly(solubility$X3,i),data = solubility
 mse_all[i]=mean((y-predict(M1,solubility))[-train]^2)

}
mse_all
```

```
##  [1] 8.491628e-03 5.536278e-02 4.789801e-01 2.309584e+02
##  [5] 8.419888e+00 6.190784e+03 7.432226e+02 1.475817e+03
##  [9] 9.661733e+02 9.661733e+02
```

```
#for x2
n=10
mse_x2=numeric(n)
for (i in 1:10) {
  M2=lm(y~poly(X2,i),data = solubility,subset = train)
 mse_x2[i]=mean((y-predict(M2,solubility))[-train]^2)

}
mse_x2
```

```
##  [1] 2.758859e-02 7.934168e-02 1.215468e+00 4.410867e-01
##  [5] 1.655272e+01 2.142517e+02 2.333635e+05 2.637888e+07
##  [9] 7.245153e+06 3.124617e+10
```

```
#for x3
n=10
mse_x3=numeric(n)
for (i in 1:10) {
  M3=lm(y~poly(x3,i),data = solubility,subset = train)
 mse_x3[i]=mean((y-predict(M3,solubility))[-train]^2)

}
mse_x3
```

```
##  [1] 1.403843e-02 3.992314e-01 4.320913e+00 1.181133e+00
##  [5] 1.201184e+03 9.823700e+04 2.356199e+05 6.581398e+07
##  [9] 1.231602e+07 2.789936e+11
```

```
mse=cbind(mse_x1,mse_x2,mse_x3)
mse
```

```
##             mse_x1       mse_x2       mse_x3
##  [1,] 1.070544e-02 2.758859e-02 1.403843e-02
##  [2,] 1.732388e-01 7.934168e-02 3.992314e-01
```

```
##  [3,] 5.258402e+00 1.215468e+00 4.320913e+00
##  [4,] 4.868209e+02 4.410867e-01 1.181133e+00
##  [5,] 3.974368e+03 1.655272e+01 1.201184e+03
##  [6,] 2.975607e+06 2.142517e+02 9.823700e+04
##  [7,] 1.295035e+04 2.333635e+05 2.356199e+05
##  [8,] 3.441342e+02 2.637888e+07 6.581398e+07
##  [9,] 2.902088e+00 7.245153e+06 1.231602e+07
## [10,] 4.541856e+00 3.124617e+10 2.789936e+11
```

• Model containing all the variables performs better for the quadratic function as it has the lowest MSE ,after that MSE increases.

• MSE for the model containing x2 variable also performs better for the quadratic function than the other functions.

• MSE for the model having x3 variable performs better for the seventh function than the other functions.

• MSE for the model having only x3 variable is best than other models as its MSE is lowest .

2 Leave One Out Cross Validation

```
solubility<-read.csv("solubility.csv")
head(solubility)
```

```
##   Observation.Number     y  X1  X2  X3
## 1                  1 0.222 7.3 0.0 0.0
## 2                  2 0.395 8.7 0.0 0.3
## 3                  3 0.422 8.8 0.7 1.0
## 4                  4 0.437 8.1 4.0 0.2
## 5                  5 0.428 9.0 0.5 1.0
## 6                  6 0.467 8.7 1.5 2.8
```

```
attach(solubility)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     y
```

```
## The following objects are masked from solubility (pos = 3):
##
##     Observation.Number, X1, X2, X3, y
```

```
## The following objects are masked from solubility (pos = 4):
##
##     Observation.Number, X1, X2, X3, y
```

```
## The following objects are masked from solubility (pos = 5):
##
##     Observation.Number, X1, X2, X3, y
```

```
## The following objects are masked from solubility (pos = 6):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 7):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 8):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 9):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 10):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 11):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 12):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 13):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 14):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 16):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 18):
##
##     Observation.Number, X1, X2, X3, y

## The following objects are masked from solubility (pos = 19):
##
##     Observation.Number, X1, X2, X3, y
```

```
library(boot)
loo.fit=glm(y~X1+X2+X3,data = solubility)
coef(loo.fit)
```

```
## (Intercept)          X1          X2          X3
## -0.36930592  0.08651029  0.02441173 -0.02857704
```

```
looVald_err=cv.glm(solubility,loo.fit) #MSE
looVald_err$delta[1]
```

```
## [1] 0.006711124
```

```
looVald=rep(0,10)
for (k in 1:10) {
  glm=glm(solubility$y~poly(solubility$X1,k)+poly(solubility$X2,k)+poly(solubility$X3,k),data = solubili
  looVald[k]=cv.glm(solubility,glm)$delta[1] #MSE

}

looVald
```

```
##  [1] 0.05133921 0.05229471 0.05391223 0.05423141
##  [5] 0.05446727 0.05477185 0.05492516 0.05499836
##  [9] 0.05500438 0.05500438
```

```
looVald_x2=rep(0,10)
for (l in 1:10) {
  glm2=glm(solubility$y~poly(solubility$X2,l),data = solubility)
  looVald_x2[l]=cv.glm(solubility,glm2)$delta[1] #MSE

}

looVald_x2
```

```
##  [1] 0.03350163 0.03351137 0.03426175 0.03468542
##  [5] 0.03938463 0.04068638 0.04164281 0.04195357
##  [9] 0.04390857 0.04491207
```

```
looVald_x3=rep(0,10)
for (m in 1:10) {
  glm3=glm(solubility$y~poly(solubility$X3,l),data = solubility)
  looVald_x3[l]=cv.glm(solubility,glm3)$delta[1] #MSE

}

looVald_x3
```

```
##  [1] 0.00000000 0.00000000 0.00000000 0.00000000
##  [5] 0.00000000 0.00000000 0.00000000 0.00000000
##  [9] 0.00000000 0.05386692
```

- There is no sharp drop in the estimated test MSE ,so there is no clear improvement from using higher-order polynomials.

3 K-Fold Cross Validation

```
kfold_cv.err<-rep(0,10)
for (i in 1:10) {
  kfold.fit<-glm(solubility$y~poly(solubility$X1,i)+poly(solubility$X2,i)+poly(solubility$X3,i),data = s

  kfold_cv.err[i]<-cv.glm(solubility,kfold.fit,K=10)$delta[1]
}

kfold_cv.err
```

```
##  [1] 0.04863720 0.05024303 0.04915868 0.05209015
##  [5] 0.05119081 0.05050192 0.05025271 0.05328011
##  [9] 0.05129763 0.05121380
```

```
MSE=cbind(looVald,kfold_cv.err)
MSE
```

```
##          looVald kfold_cv.err
##  [1,] 0.05133921   0.04863720
##  [2,] 0.05229471   0.05024303
##  [3,] 0.05391223   0.04915868
##  [4,] 0.05423141   0.05209015
##  [5,] 0.05446727   0.05119081
##  [6,] 0.05477185   0.05050192
##  [7,] 0.05492516   0.05025271
##  [8,] 0.05499836   0.05328011
##  [9,] 0.05500438   0.05129763
## [10,] 0.05500438   0.05121380
```

- The computational time is shorter than that of LOOCV .

- MSE of the quadratic ,cubic or higher order polynomial has lower MSE but not as much that should be considered so only linear variables would be considered in our model.