Splines and Cross Validation

STAT 380

# Loading the BostonHousing Data

1. Using the read.csv() command: Read in the R worksopace as the name ‘Daten’ Note that the names are case sensitive. that is ‘Daten’ is not same as ‘daten’.
2. Apply the basic functions on the data.frame ‘Daten’ to check its structure.

#########################################################################  
# STAT380: Statistical Machine Learning  
# Content: Dataset Boston housing   
#########################################################################  
  
#Daten<-read.csv("")  
  
Daten<-read.csv("https://raw.githubusercontent.com/subhadippal2019/STAT380UAEU/main/BostonHousing.csv")  
#Daten  
  
  
# Additional information for a quick check to identify whether the data is loaded appropriately  
head(Daten, 6) # shows the first 6 rows from the data.

## CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX PTRATIO LSTAT MEDV  
## 1 0.00632 18 2.31 0 0.538 6.575 65.2 4.0900 1 296 15.3 4.98 24.0  
## 2 0.02731 0 7.07 0 0.469 6.421 78.9 4.9671 2 242 17.8 9.14 21.6  
## 3 0.02729 0 7.07 0 0.469 7.185 61.1 4.9671 2 242 17.8 4.03 34.7  
## 4 0.03237 0 2.18 0 0.458 6.998 45.8 6.0622 3 222 18.7 2.94 33.4  
## 5 0.06905 0 2.18 0 0.458 7.147 54.2 6.0622 3 222 18.7 5.33 36.2  
## 6 0.02985 0 2.18 0 0.458 6.430 58.7 6.0622 3 222 18.7 5.21 28.7  
## CAT..MEDV  
## 1 0  
## 2 0  
## 3 1  
## 4 1  
## 5 1  
## 6 0

str(Daten) # provides structure of an R object in general. In this case it will show the names of the variables inside it.

## 'data.frame': 506 obs. of 14 variables:  
## $ CRIM : num 0.00632 0.02731 0.02729 0.03237 0.06905 ...  
## $ ZN : num 18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...  
## $ INDUS : num 2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...  
## $ CHAS : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ NOX : num 0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...  
## $ RM : num 6.58 6.42 7.18 7 7.15 ...  
## $ AGE : num 65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...  
## $ DIS : num 4.09 4.97 4.97 6.06 6.06 ...  
## $ RAD : int 1 2 2 3 3 3 5 5 5 5 ...  
## $ TAX : int 296 242 242 222 222 222 311 311 311 311 ...  
## $ PTRATIO : num 15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...  
## $ LSTAT : num 4.98 9.14 4.03 2.94 5.33 ...  
## $ MEDV : num 24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...  
## $ CAT..MEDV: int 0 0 1 1 1 0 0 0 0 0 ...

summary(Daten) # This commands provides the summary of all the variables of the dataset.

## CRIM ZN INDUS CHAS   
## Min. : 0.00632 Min. : 0.00 Min. : 0.46 Min. :0.00000   
## 1st Qu.: 0.08205 1st Qu.: 0.00 1st Qu.: 5.19 1st Qu.:0.00000   
## Median : 0.25651 Median : 0.00 Median : 9.69 Median :0.00000   
## Mean : 3.61352 Mean : 11.36 Mean :11.14 Mean :0.06917   
## 3rd Qu.: 3.67708 3rd Qu.: 12.50 3rd Qu.:18.10 3rd Qu.:0.00000   
## Max. :88.97620 Max. :100.00 Max. :27.74 Max. :1.00000   
## NOX RM AGE DIS   
## Min. :0.3850 Min. :3.561 Min. : 2.90 Min. : 1.130   
## 1st Qu.:0.4490 1st Qu.:5.886 1st Qu.: 45.02 1st Qu.: 2.100   
## Median :0.5380 Median :6.208 Median : 77.50 Median : 3.207   
## Mean :0.5547 Mean :6.285 Mean : 68.57 Mean : 3.795   
## 3rd Qu.:0.6240 3rd Qu.:6.623 3rd Qu.: 94.08 3rd Qu.: 5.188   
## Max. :0.8710 Max. :8.780 Max. :100.00 Max. :12.127   
## RAD TAX PTRATIO LSTAT   
## Min. : 1.000 Min. :187.0 Min. :12.60 Min. : 1.73   
## 1st Qu.: 4.000 1st Qu.:279.0 1st Qu.:17.40 1st Qu.: 6.95   
## Median : 5.000 Median :330.0 Median :19.05 Median :11.36   
## Mean : 9.549 Mean :408.2 Mean :18.46 Mean :12.65   
## 3rd Qu.:24.000 3rd Qu.:666.0 3rd Qu.:20.20 3rd Qu.:16.95   
## Max. :24.000 Max. :711.0 Max. :22.00 Max. :37.97   
## MEDV CAT..MEDV   
## Min. : 5.00 Min. :0.000   
## 1st Qu.:17.02 1st Qu.:0.000   
## Median :21.20 Median :0.000   
## Mean :22.53 Mean :0.166   
## 3rd Qu.:25.00 3rd Qu.:0.000   
## Max. :50.00 Max. :1.000

# Data Partition

1. Split data in two partitions (80% Training, 20% Validation)
2. set.seed(10)
3. Use the function create Data Partition from `caret’ package.
4. Read the help manual for the function `createDataPartition’ The corresponding command is ‘?createDataPartition’ OR ‘help(createDataPartition)’

#########################################################################  
 # 2. Data Partitioning   
 #########################################################################  
   
 # Split data in three partitions (80% Training, 20% Validation)  
   
 library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

#install.packages('caret') # Use this command to install the package, if case the package is not installed and you get an error "rror in library(caret) : there is no package called ‘caret’"  
 set.seed(10) # An optional argument butr sometimes convinient to set the see to reproduce the result.   
 #We are going to use the createDataPartition function. Hence see the documentation on the function  
 # help(createDataPartition)

library(caret)  
  
createDataPartition\_alt<-function(y,p = 0.5, list=FALSE){  
 n=length(y)  
 Train\_size=round(n\*p)  
 sel\_sample=sample(x = 1:n,size =Train\_size ,replace = FALSE)  
 return(sel\_sample)  
}

Daten=na.omit(Daten)  
 #inTrain = createDataPartition(Daten$CRIM, p = 0.8, list = FALSE)  
 inTrain = createDataPartition\_alt(Daten$CRIM, p = 0.8, list = FALSE)  
 train = Daten[inTrain, ]  
 dim(train)

## [1] 405 14

dim(Daten)

## [1] 506 14

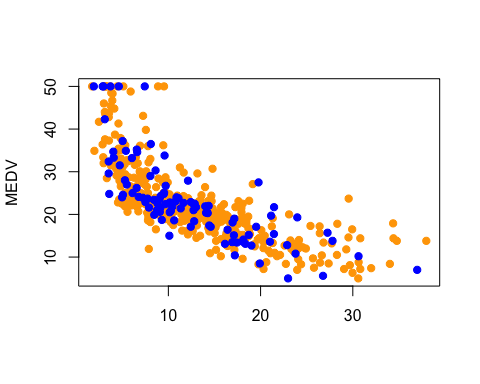
## Creating the Testing Set

1. Create the Testing set ‘test’. Not ethat test set will have all the data rows that are not included in the trainig set.
2. Plot the variable ‘MEDV’ available in the ‘train’ and ‘test’ dataset and plot the points in a different color
3. Put the legend command to see what changes does it make to the plot.

test = Daten[-inTrain,]  
 dim(test)

## [1] 101 14

plot(train$LSTAT, train$MEDV, col="orange", pch=19 , ylab="MEDV", xlab="")  
 points(test$LSTAT, test$MEDV,col="blue", pch=19)  
 legend(280, 47, legend=c("train$MEDV", "test$MEDV"),  
 col=c("orange" ,"blue"), lty=1:2, cex=0.8)



# Fitting a Polynomial

1. We fit in the training dataset and check its performance from the testing dataset Degree of the polynomial is 2 in the following example. I.e. the function of the type will be considered.

model\_polynomial <- lm(MEDV ~ poly(LSTAT, 2, raw = TRUE), data= train)  
  
  
#model\_linear = lm(MEDV ~ LSTAT, data = train);   
summary(model\_polynomial)

##   
## Call:  
## lm(formula = MEDV ~ poly(LSTAT, 2, raw = TRUE), data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -15.0806 -3.8866 -0.3915 2.3088 25.5758   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 42.432448 0.965929 43.93 <2e-16 \*\*\*  
## poly(LSTAT, 2, raw = TRUE)1 -2.301207 0.136649 -16.84 <2e-16 \*\*\*  
## poly(LSTAT, 2, raw = TRUE)2 0.043186 0.004116 10.49 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 5.445 on 402 degrees of freedom  
## Multiple R-squared: 0.6392, Adjusted R-squared: 0.6374   
## F-statistic: 356.1 on 2 and 402 DF, p-value: < 2.2e-16

#Make predictions  
library(forecast)

## Warning: package 'forecast' was built under R version 4.0.5

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

predictions\_test = predict(object =model\_polynomial,newdata= test )  
  
 #MOdel performance on the Testing Data  
 data.frame(RMSE = RMSE(predictions\_test, test$MEDV), R2 = R2(predictions\_test, test$MEDV))

## RMSE R2  
## 1 5.839985 0.6488598

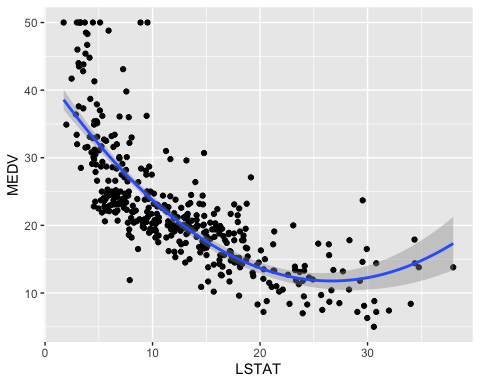
accuracy(predictions\_test, test$MEDV)

## ME RMSE MAE MPE MAPE  
## Test set 0.5292977 5.839985 4.275274 -4.846819 20.31902

#MOdel performance on the Training Data   
 predictions\_train = predict(object =model\_polynomial,newdata= train )  
 data.frame(RMSE = RMSE(predictions\_train, train$MEDV), R2 = R2(predictions\_train, train$MEDV))

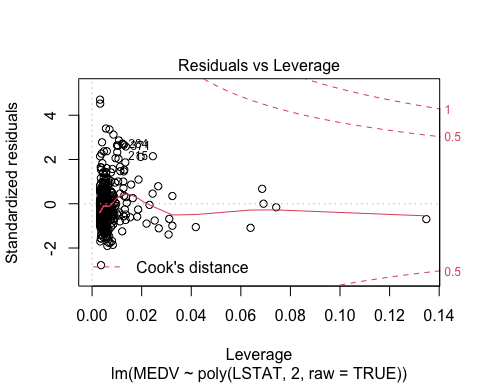
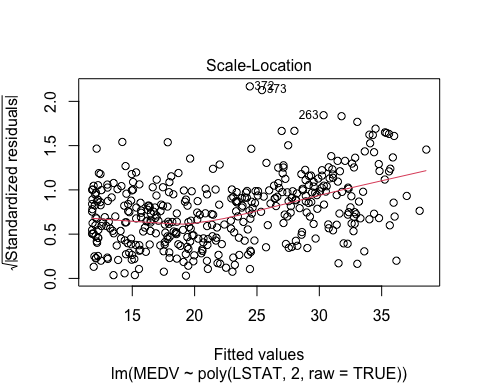
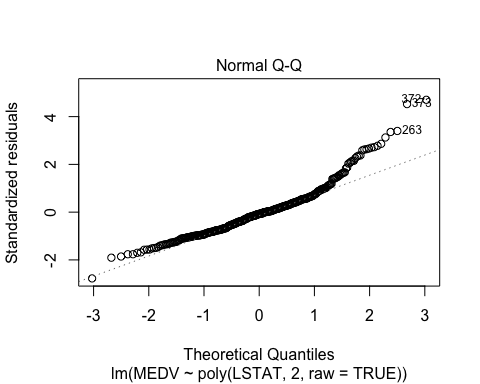
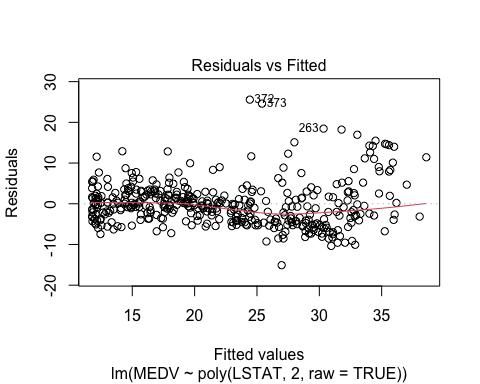
## RMSE R2  
## 1 5.424907 0.6392125

plt<-ggplot(train, aes(LSTAT, MEDV) ) + geom\_point() +stat\_smooth(method = lm, formula = y ~ poly(x, 2, raw =TRUE))  
plt



## Checking the model Assumptions:

plot(model\_polynomial)



# Fitting a Regression Splines

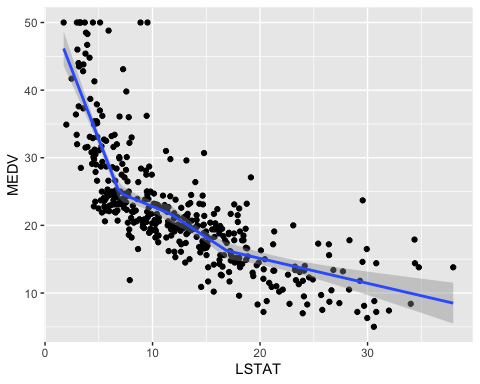
## 1) Consider Fiting a piecewise Line

Step1: Identify the regions to put the knots Step2: Fit a spline Step3: plot the results

#install.packages("splines")  
library(splines)  
 # Build the model  
knots <- quantile(train$LSTAT, p = c(0.25, .5, 0.75))  
model\_ss1 <- lm (MEDV ~ bs(LSTAT, knots = knots, degree=1), data =train)  
summary(model\_ss1)

##   
## Call:  
## lm(formula = MEDV ~ bs(LSTAT, knots = knots, degree = 1), data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -12.2749 -3.1638 -0.8243 2.4137 26.9017   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 46.159 1.290 35.77 <2e-16 \*\*\*  
## bs(LSTAT, knots = knots, degree = 1)1 -21.358 1.686 -12.67 <2e-16 \*\*\*  
## bs(LSTAT, knots = knots, degree = 1)2 -24.451 1.385 -17.65 <2e-16 \*\*\*  
## bs(LSTAT, knots = knots, degree = 1)3 -29.901 1.449 -20.64 <2e-16 \*\*\*  
## bs(LSTAT, knots = knots, degree = 1)4 -37.660 1.996 -18.87 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 5.158 on 400 degrees of freedom  
## Multiple R-squared: 0.6779, Adjusted R-squared: 0.6747   
## F-statistic: 210.5 on 4 and 400 DF, p-value: < 2.2e-16

ggplot(train, aes(LSTAT, MEDV) ) + geom\_point() +  
stat\_smooth(method = lm, formula = y ~ splines::bs(x, knots = knots, degree=1))

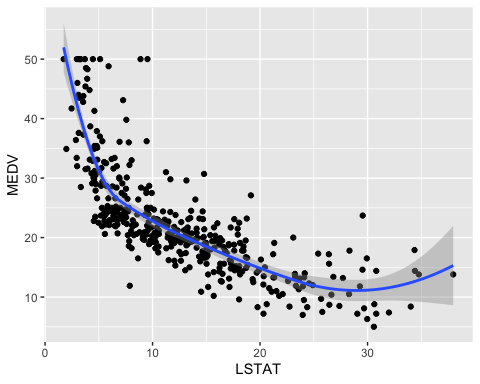


## 1) Consider Fiting a piecewise polynomial of degree 2

#install.packages("splines")  
 # Build the model  
knots <- quantile(train$LSTAT, p = c(0.25, .5, 0.9))  
model\_ss2 <- lm (MEDV ~ bs(LSTAT, knots = knots, degree=2), data =train)  
summary(model\_ss2)

##   
## Call:  
## lm(formula = MEDV ~ bs(LSTAT, knots = knots, degree = 2), data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -14.9561 -3.1347 -0.7419 2.0374 26.7507   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 51.980 2.123 24.488 < 2e-16 \*\*\*  
## bs(LSTAT, knots = knots, degree = 2)1 -22.514 2.928 -7.688 1.17e-13 \*\*\*  
## bs(LSTAT, knots = knots, degree = 2)2 -28.709 2.066 -13.898 < 2e-16 \*\*\*  
## bs(LSTAT, knots = knots, degree = 2)3 -35.761 2.429 -14.722 < 2e-16 \*\*\*  
## bs(LSTAT, knots = knots, degree = 2)4 -43.532 2.681 -16.236 < 2e-16 \*\*\*  
## bs(LSTAT, knots = knots, degree = 2)5 -36.677 4.060 -9.034 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 5.109 on 399 degrees of freedom  
## Multiple R-squared: 0.6848, Adjusted R-squared: 0.6808   
## F-statistic: 173.4 on 5 and 399 DF, p-value: < 2.2e-16

ggplot(train, aes(LSTAT, MEDV) ) + geom\_point() +  
stat\_smooth(method = lm, formula = y ~ splines::bs(x, knots = knots, degree=2))

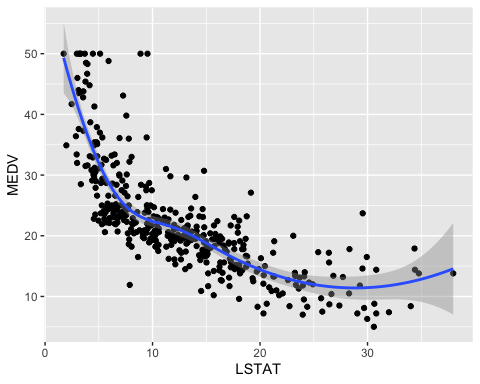


## 1) Consider Fiting a piecewise polynomial of degree 3 ( most popular choice also the default option)

#install.packages("splines")  
library(splines)  
 # Build the model  
knots <- quantile(train$LSTAT, p = c(0.25, .5, 0.75))  
model\_ss3 <- lm (MEDV ~ bs(LSTAT, knots = knots), data =train)  
summary(model\_ss3)

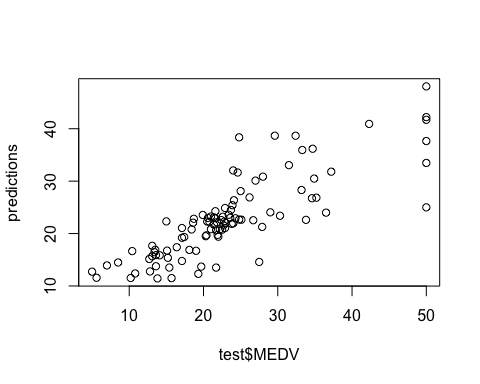
##   
## Call:  
## lm(formula = MEDV ~ bs(LSTAT, knots = knots), data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -12.7884 -3.0077 -0.7195 2.0113 27.3480   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 49.324 2.958 16.677 < 2e-16 \*\*\*  
## bs(LSTAT, knots = knots)1 -11.488 4.369 -2.629 0.00889 \*\*   
## bs(LSTAT, knots = knots)2 -26.715 2.786 -9.589 < 2e-16 \*\*\*  
## bs(LSTAT, knots = knots)3 -26.955 3.316 -8.128 5.58e-15 \*\*\*  
## bs(LSTAT, knots = knots)4 -38.776 3.446 -11.251 < 2e-16 \*\*\*  
## bs(LSTAT, knots = knots)5 -39.600 4.743 -8.349 1.15e-15 \*\*\*  
## bs(LSTAT, knots = knots)6 -34.760 4.759 -7.305 1.53e-12 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 5.125 on 398 degrees of freedom  
## Multiple R-squared: 0.6836, Adjusted R-squared: 0.6788   
## F-statistic: 143.3 on 6 and 398 DF, p-value: < 2.2e-16

ggplot(train, aes(LSTAT, MEDV) ) + geom\_point() +  
stat\_smooth(method = lm, formula = y ~ splines::bs(x, knots = knots))



## Prediction Based on the 3 degree polynomial Fit.

#predictions <- modelss %>% predict(test)  
predictions <- predict(model\_ss3, test)  
plot(test$MEDV , predictions)



# Model performance  
data.frame(  
RMSE = RMSE(predictions, test$MEDV),  
R2 = R2(predictions, test$MEDV))

## RMSE R2  
## 1 5.559719 0.6789511

# Cross Validation

set.seed(11)  
fit<-glm(MEDV~CRIM+RM+PTRATIO,data=Daten)  
  
library(boot)

##   
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':  
##   
## melanoma

# Leave-one-out cross-validation  
cv\_one\_err<-cv.glm(Daten,fit)  
cv\_one\_err$delta

## [1] 35.00064 34.99989

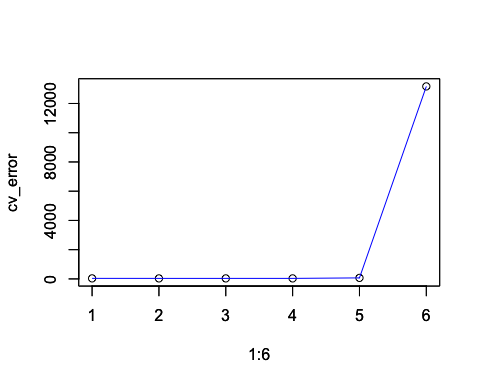
# 5 fold Cross Validation  
cv\_5\_err<-cv.glm(Daten,fit,K=5)  
cv\_5\_err$delta

## [1] 36.09778 35.89034

cv\_error<-NULL  
  
 for(i in 1:6){  
 fit\_poly<-glm(MEDV~poly(CRIM,degree=i)+RM+PTRATIO,data=Daten)  
 cv\_error[i]<-cv.glm(Daten,fit\_poly,K=5)$delta[1]  
 }  
 cv\_error

## [1] 36.50394 34.61901 35.16963 34.97346 69.66155 13168.04656

#[1] 34.708 34.813 35.076 34.423 41.874 51.530  
  
plot(1:6, cv\_error)  
par(new=TRUE)  
plot(1:6, cv\_error, type="l", col="blue")



## Preparation for the next class

#install.packages("glmnet"  
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.0-2