Ridge regression

Soundarya G

26/02/2021

**1.Load mtcars dataset**

data(mtcars)

head(mtcars)

## mpg cyl disp hp drat wt qsec vs am gear carb  
## Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4  
## Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4  
## Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1  
## Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1  
## Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2  
## Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

**Inference:** The mtcars dataset is loaded and first 6 data are viewed.

**2.install ridge and glmnet packages**

library(ridge)

## Warning: package 'ridge' was built under R version 4.0.3

library(glmnet)

## Warning: package 'glmnet' was built under R version 4.0.3

## Loading required package: Matrix

## Loaded glmnet 4.1

**Inference:** The ridge and glmnet packages are installed and libraries are loaded.

**3.Perform the exploratory data analysis**

#Preprocessing  
df = mtcars  
str(df)

## 'data.frame': 32 obs. of 11 variables:  
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...  
## $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...  
## $ disp: num 160 160 108 258 360 ...  
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...  
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...  
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...  
## $ qsec: num 16.5 17 18.6 19.4 17 ...  
## $ vs : num 0 0 1 1 0 1 0 1 1 1 ...  
## $ am : num 1 1 1 0 0 0 0 0 0 0 ...  
## $ gear: num 4 4 4 3 3 3 3 4 4 4 ...  
## $ carb: num 4 4 1 1 2 1 4 2 2 4 ...

summary(df)

## mpg cyl disp hp   
## Min. :10.40 Min. :4.000 Min. : 71.1 Min. : 52.0   
## 1st Qu.:15.43 1st Qu.:4.000 1st Qu.:120.8 1st Qu.: 96.5   
## Median :19.20 Median :6.000 Median :196.3 Median :123.0   
## Mean :20.09 Mean :6.188 Mean :230.7 Mean :146.7   
## 3rd Qu.:22.80 3rd Qu.:8.000 3rd Qu.:326.0 3rd Qu.:180.0   
## Max. :33.90 Max. :8.000 Max. :472.0 Max. :335.0   
## drat wt qsec vs   
## Min. :2.760 Min. :1.513 Min. :14.50 Min. :0.0000   
## 1st Qu.:3.080 1st Qu.:2.581 1st Qu.:16.89 1st Qu.:0.0000   
## Median :3.695 Median :3.325 Median :17.71 Median :0.0000   
## Mean :3.597 Mean :3.217 Mean :17.85 Mean :0.4375   
## 3rd Qu.:3.920 3rd Qu.:3.610 3rd Qu.:18.90 3rd Qu.:1.0000   
## Max. :4.930 Max. :5.424 Max. :22.90 Max. :1.0000   
## am gear carb   
## Min. :0.0000 Min. :3.000 Min. :1.000   
## 1st Qu.:0.0000 1st Qu.:3.000 1st Qu.:2.000   
## Median :0.0000 Median :4.000 Median :2.000   
## Mean :0.4062 Mean :3.688 Mean :2.812   
## 3rd Qu.:1.0000 3rd Qu.:4.000 3rd Qu.:4.000   
## Max. :1.0000 Max. :5.000 Max. :8.000

**Inference:** The min,max,mean,median and quantile values of each columns along with the class they belong to is given by the summary of the mtcars data.

#Checking for missing values.  
colSums(is.na(df))

## mpg cyl disp hp drat wt qsec vs am gear carb   
## 0 0 0 0 0 0 0 0 0 0 0

**Inference:** There are no missing values.

#Checking for Empty Values  
colSums(df=='')

## mpg cyl disp hp drat wt qsec vs am gear carb   
## 0 0 0 0 0 0 0 0 0 0 0

**Inference:** There are no Empty values.

#Checking for Duplicate values  
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.3

## -- Attaching packages ----------------------------------------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.0.3 v dplyr 1.0.4  
## v tidyr 1.1.2 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.3

## Warning: package 'tidyr' was built under R version 4.0.3

## Warning: package 'readr' was built under R version 4.0.3

## Warning: package 'dplyr' was built under R version 4.0.3

## Warning: package 'forcats' was built under R version 4.0.3

## -- Conflicts -------------------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x tidyr::expand() masks Matrix::expand()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x tidyr::pack() masks Matrix::pack()  
## x tidyr::unpack() masks Matrix::unpack()

duplicated(df)

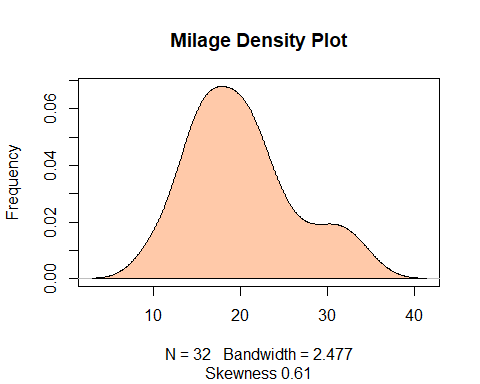
## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

**Inference:** There are no Duplicate values.

#Checking Normality of Response Variable  
  
library(e1071)

## Warning: package 'e1071' was built under R version 4.0.3

plot(density(df$mpg), main = "Milage Density Plot", ylab="Frequency", sub=paste("Skewness",round(e1071::skewness(df$mpg),2)))  
polygon(density(df$mpg), col='#FFC9A9')



**Inference:** Slightly Right Skweked, which implies most of the values are posititve in nature.

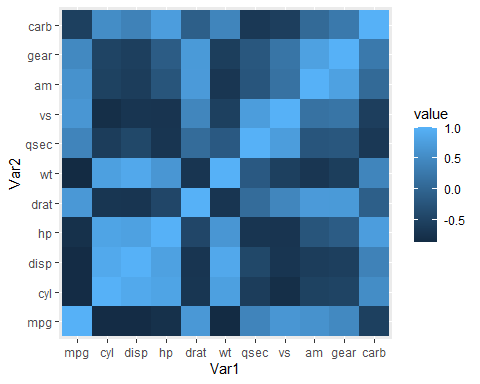
#Correlation Heat Map  
library(ggplot2)  
library(reshape2)

## Warning: package 'reshape2' was built under R version 4.0.3

##   
## Attaching package: 'reshape2'

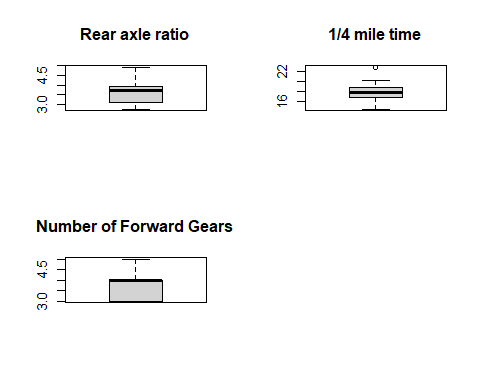
## The following object is masked from 'package:tidyr':  
##   
## smiths

cormat <- round(cor(df),2)  
melted\_cormat <- melt(cormat)  
ggplot(data = melted\_cormat, aes(x=Var1, y=Var2, fill=value)) +   
 geom\_tile()



**Inference:**It is evident that most of the variables possess a high correlation with each other, thus we can assume multicollinearity is present.

#Checking for Outliers in highly positive correlated values with Milage  
par(mfrow=c(2,2))  
boxplot(df$drat, main = "Rear axle ratio")  
boxplot(df$qsec, main = "1/4 mile time")  
boxplot(df$gear, main = "Number of Forward Gears")



**Inference:** There is an outlier found in qsec(1/4 mile time).

#Building initial model  
X = model.matrix(mpg~. , mtcars)[,-1]  
Y = mtcars$mpg  
  
#Splitting the data  
set.seed(57)  
  
trainingRow <- sample(1:nrow(df), 0.7\*nrow(df))  
trainset <- df[trainingRow,]  
testset <- df[-trainingRow,]  
  
lrm <- lm(trainset$mpg~.,data=trainset)  
  
summary(lrm)

##   
## Call:  
## lm(formula = trainset$mpg ~ ., data = trainset)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -3.1759 -1.4218 -0.7548 1.0168 4.3028   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 8.55175 25.48468 0.336 0.744  
## cyl -0.07409 1.35897 -0.055 0.957  
## disp 0.01402 0.02943 0.476 0.643  
## hp -0.04564 0.03718 -1.227 0.245  
## drat 1.01463 2.83182 0.358 0.727  
## wt -3.66520 2.79708 -1.310 0.217  
## qsec 1.22513 0.93256 1.314 0.216  
## vs -1.03224 2.89376 -0.357 0.728  
## am 4.89791 2.80389 1.747 0.108  
## gear -0.76347 2.07162 -0.369 0.719  
## carb 1.00937 1.36147 0.741 0.474  
##   
## Residual standard error: 3.006 on 11 degrees of freedom  
## Multiple R-squared: 0.8924, Adjusted R-squared: 0.7945   
## F-statistic: 9.12 on 10 and 11 DF, p-value: 0.0005314

library(car)

## Warning: package 'car' was built under R version 4.0.3

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.0.3

##   
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':  
##   
## recode

## The following object is masked from 'package:purrr':  
##   
## some

vif(lrm)

## cyl disp hp drat wt qsec vs am   
## 14.567223 35.146018 19.377286 5.983737 20.942264 7.286081 4.928560 4.627183   
## gear carb   
## 4.922279 12.942380

**Inference:** All the values are above 5, there is strong multi-collinearity present.

MLR\_pred <- predict(lrm,testset)  
compare <- cbind(actual=testset$mpg,MLR\_pred)  
compare

## actual MLR\_pred  
## Mazda RX4 21.0 25.73138  
## Mazda RX4 Wag 21.0 25.48283  
## Hornet Sportabout 18.7 16.18709  
## Merc 450SE 16.4 13.86304  
## Merc 450SL 17.3 15.35424  
## Dodge Challenger 15.5 15.86656  
## Camaro Z28 13.3 12.02072  
## Pontiac Firebird 19.2 15.22923  
## Lotus Europa 30.4 25.48280  
## Ferrari Dino 19.7 21.79969

mean (apply(compare, 1, min)/apply(compare, 1, max))

## [1] 0.8654496

RMSE = sqrt(mean((testset$mpg-MLR\_pred)^2))  
RMSE# calculate accuracy

## [1] 3.242592

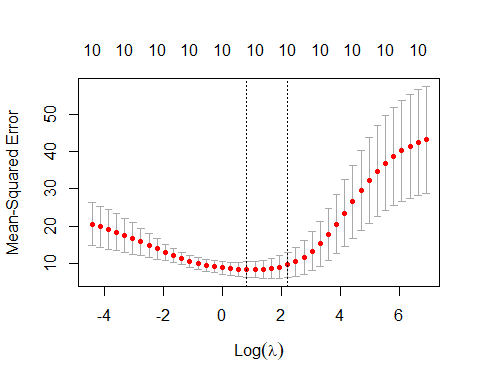
**Inference:** Accuracy is only 81%, which is not very efficient.

**4.Choose optimum lamba value**

#Creating a sequence with an interval of -0.12  
lambda\_seq = 10^seq(3, -2, by = -.12)  
  
# Using cross validation glmnet  
ridge\_model1 = cv.glmnet(X[trainingRow,], Y[trainingRow],alpha = 0, type.measure = "mse", lambda = lambda\_seq, nfolds = 5)  
  
# Best lambda value  
best\_lam = ridge\_model1$lambda.min  
best\_lam

## [1] 2.290868

plot(ridge\_model1)



**Inference:** Optimum lamba value choosed and plotted.

**5.Extract the model using k-cross validation**

best\_fit <- ridge\_model1$glmnet.fit  
head(best\_fit)

## $a0  
## s0 s1 s2 s3 s4 s5 s6 s7   
## 20.334559 20.292781 20.239225 20.171147 20.085525 19.979298 19.849725 19.694961   
## s8 s9 s10 s11 s12 s13 s14 s15   
## 19.514784 19.311363 19.089707 18.858175 18.627489 18.408886 18.213989 18.048764   
## s16 s17 s18 s19 s20 s21 s22 s23   
## 17.916799 17.809536 17.713541 17.605253 17.453561 17.211267 16.889578 16.443811   
## s24 s25 s26 s27 s28 s29 s30 s31   
## 15.867262 15.168862 14.373356 13.507767 12.617859 11.748121 10.943290 10.235891   
## s32 s33 s34 s35 s36 s37 s38 s39   
## 9.652625 9.195744 8.862197 8.633752 8.489987 8.407891 8.371597 8.363156   
## s40 s41   
## 8.371917 8.390624   
##   
## $beta  
## 10 x 42 sparse Matrix of class "dgCMatrix"

## [[ suppressing 42 column names 's0', 's1', 's2' ... ]]

##   
## cyl -0.0194178748 -0.0252966251 -0.0328380896 -0.0424341586 -0.0545182791  
## disp -0.0002758091 -0.0003593572 -0.0004665682 -0.0006030428 -0.0007749987  
## hp -0.0004186479 -0.0005454341 -0.0007081107 -0.0009151644 -0.0011760196  
## drat 0.0533808300 0.0695996484 0.0904465457 0.1170416036 0.1506476225  
## wt -0.0334154670 -0.0435648435 -0.0566082128 -0.0732447062 -0.0942618997  
## qsec 0.0099463820 0.0129456398 0.0167847447 0.0216558137 0.0277673440  
## vs 0.0552584274 0.0719235805 0.0932558041 0.1203216130 0.1542757009  
## am 0.0480936150 0.0627816340 0.0817150371 0.1059602526 0.1367494081  
## gear 0.0231786989 0.0302151718 0.0392546475 0.0507776494 0.0653217181  
## carb -0.0151921901 -0.0197946406 -0.0257010946 -0.0332208008 -0.0426979387  
##   
## cyl -0.0695388873 -0.087907973 -0.109928221 -0.135698629 -0.165015766  
## disp -0.0009888889 -0.001250704 -0.001564942 -0.001933281 -0.002353179  
## hp -0.0015004572 -0.001897588 -0.002374333 -0.002933503 -0.003571833  
## drat 0.1926063151 0.244220147 0.306569016 0.380269080 0.465207714  
## wt -0.1204958347 -0.152757675 -0.191720764 -0.237773900 -0.290863548  
## qsec 0.0353274579 0.044516063 0.055444978 0.068109946 0.082345031  
## vs 0.1962638343 0.247258973 0.307821869 0.377800951 0.456021858  
## am 0.1754431571 0.223451530 0.282103430 0.352467645 0.435148965  
## gear 0.0834498460 0.105691965 0.132453336 0.163890040 0.199763730  
## carb -0.0544910930 -0.068936941 -0.086296270 -0.106686059 -0.130010072  
##   
## cyl -0.197283638 -0.231536306 -0.266464233 -0.300532751 -0.332346390  
## disp -0.002816699 -0.003310496 -0.003816427 -0.004313505 -0.004781762  
## hp -0.004278610 -0.005035651 -0.005819007 -0.006602703 -0.007362891  
## drat 0.560329707 0.663541512 0.771838464 0.881686019 0.989424926  
## wt -0.350375086 -0.415083181 -0.483257506 -0.552919185 -0.622047734  
## qsec 0.097799533 0.113949489 0.130172146 0.145876393 0.160608946  
## vs 0.540065290 0.626210735 0.709690715 0.785227191 0.847558044  
## am 0.530113606 0.636590523 0.753147026 0.877942427 1.008998300  
## gear 0.239303462 0.281113209 0.323170856 0.362934380 0.397554449  
## carb -0.155910631 -0.183762893 -0.212736212 -0.241908068 -0.270379880  
##   
## cyl -0.360528450 -0.384297972 -0.402843612 -0.415951073 -0.423452517  
## disp -0.005203051 -0.005565010 -0.005860181 -0.006085898 -0.006242758  
## hp -0.008082416 -0.008750542 -0.009369868 -0.009946048 -0.010492122  
## drat 1.091995922 1.186915651 1.273366396 1.351086594 1.420900908  
## wt -0.689079862 -0.752763379 -0.812840558 -0.869349465 -0.923011403  
## qsec 0.174252289 0.186977416 0.199466299 0.212655588 0.227820785  
## vs 0.892306277 0.916099663 0.917099338 0.894707619 0.849574240  
## am 1.144739301 1.284064223 1.426749223 1.573181967 1.724338051  
## gear 0.424144323 0.440141829 0.443347589 0.432389871 0.406596869  
## carb -0.297386038 -0.322313670 -0.344555506 -0.363509808 -0.378370411  
##   
## cyl -0.425167325 -0.42005450 -0.410120588 -0.393896701 -0.371281213  
## disp -0.006333105 -0.00635348 -0.006332249 -0.006253825 -0.006116423  
## hp -0.011026914 -0.01158088 -0.012176779 -0.012846804 -0.013629891  
## drat 1.484296094 1.54371974 1.598688328 1.650251164 1.697789077  
## wt -0.975109142 -1.02826126 -1.081801192 -1.139297646 -1.202947602  
## qsec 0.246460918 0.27054145 0.300209136 0.337137522 0.381754764  
## vs 0.783255381 0.69856011 0.594295394 0.475105213 0.343307682  
## am 1.881503356 2.04674980 2.218420823 2.398691570 2.587386755  
## gear 0.365963926 0.31086667 0.242390790 0.162379925 0.072948414  
## carb -0.388033773 -0.39096250 -0.385571567 -0.370499385 -0.344427489  
##   
## cyl -0.342463622 -0.30873866 -0.270348779 -0.229806214 -0.188672924  
## disp -0.005911888 -0.00561437 -0.005202658 -0.004637032 -0.003898958  
## hp -0.014564976 -0.01566547 -0.016983067 -0.018515083 -0.020283937  
## drat 1.739537496 1.77200174 1.793661659 1.801191849 1.793749442  
## wt -1.274860253 -1.35695787 -1.451703556 -1.560721178 -1.684862780  
## qsec 0.433804547 0.49221963 0.555823870 0.622550805 0.690414395  
## vs 0.201754515 0.05567978 -0.091622500 -0.234074095 -0.368759878  
## am 2.783544366 2.98484806 3.188974795 3.391538649 3.588815085  
## gear -0.023108235 -0.12124949 -0.218706221 -0.310809428 -0.395541062  
## carb -0.306557135 -0.25743708 -0.196781736 -0.126129760 -0.046146270  
##   
## cyl -0.14959444 -0.114321523 -0.0859363880 -0.0640708118 -0.049428577  
## disp -0.00296822 -0.001852804 -0.0005431506 0.0008965152 0.002426298  
## hp -0.02226644 -0.024430897 -0.0266952816 -0.0290085396 -0.031286481  
## drat 1.77054264 1.732788652 1.6811238095 1.6198149498 1.551854476  
## wt -1.82401345 -1.976241190 -2.1402540939 -2.3104968329 -2.482769028  
## qsec 0.75729715 0.821420677 0.8812996172 0.9360143581 0.984986034  
## vs -0.49177145 -0.601396374 -0.6940612888 -0.7717114703 -0.834179069  
## am 3.77642780 3.951100196 4.1088358846 4.2489768787 4.370390310  
## gear -0.47057546 -0.535317980 -0.5882799241 -0.6313053773 -0.664915963  
## carb 0.04134609 0.134470039 0.2303378572 0.3267476137 0.420873237  
##   
## cyl -0.040722148 -0.036836911 -0.036072576 -0.037993027 -0.040630461  
## disp 0.003967958 0.005458692 0.006830848 0.008074849 0.009127739  
## hp -0.033461559 -0.035471692 -0.037271014 -0.038840130 -0.040163313  
## drat 1.481742944 1.413049417 1.349248180 1.291229019 1.241674091  
## wt -2.650541028 -2.808574581 -2.951544054 -3.078901358 -3.186057739  
## qsec 1.027968753 1.064995577 1.096234180 1.122274794 1.143304513  
## vs -0.883811415 -0.922412693 -0.952499479 -0.974827201 -0.992625860  
## am 4.473871578 4.560469250 4.631924264 4.689726505 4.736066941  
## gear -0.690910987 -0.710635426 -0.725678449 -0.736618641 -0.745245241  
## carb 0.510141494 0.592324213 0.665634193 0.729630910 0.783468884  
##   
## cyl -0.04406097 -0.04759208  
## disp 0.01002471 0.01075943  
## hp -0.04126095 -0.04214877  
## drat 1.19944143 1.16467932  
## wt -3.27637387 -3.34992102  
## qsec 1.16026691 1.17361996  
## vs -1.00570722 -1.01567187  
## am 4.77257000 4.80094436  
## gear -0.75148098 -0.75619184  
## carb 0.82821911 0.86447869  
##   
## $df  
## [1] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  
## [26] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  
##   
## $dim  
## [1] 10 42  
##   
## $lambda  
## [1] 1.000000e+03 7.585776e+02 5.754399e+02 4.365158e+02 3.311311e+02  
## [6] 2.511886e+02 1.905461e+02 1.445440e+02 1.096478e+02 8.317638e+01  
## [11] 6.309573e+01 4.786301e+01 3.630781e+01 2.754229e+01 2.089296e+01  
## [16] 1.584893e+01 1.202264e+01 9.120108e+00 6.918310e+00 5.248075e+00  
## [21] 3.981072e+00 3.019952e+00 2.290868e+00 1.737801e+00 1.318257e+00  
## [26] 1.000000e+00 7.585776e-01 5.754399e-01 4.365158e-01 3.311311e-01  
## [31] 2.511886e-01 1.905461e-01 1.445440e-01 1.096478e-01 8.317638e-02  
## [36] 6.309573e-02 4.786301e-02 3.630781e-02 2.754229e-02 2.089296e-02  
## [41] 1.584893e-02 1.202264e-02  
##   
## $dev.ratio  
## [1] 0.06108987 0.07915888 0.10204834 0.13070259 0.16603686 0.20879069  
## [7] 0.25931876 0.31734285 0.38172109 0.45033252 0.52018246 0.58780504  
## [13] 0.64985860 0.70377117 0.74818554 0.78300796 0.80915562 0.82815348  
## [19] 0.84165396 0.85118415 0.85799058 0.86302353 0.86690343 0.87007810  
## [25] 0.87282648 0.87530903 0.87758983 0.87972123 0.88169272 0.88350751  
## [31] 0.88515014 0.88661178 0.88788129 0.88895987 0.88984982 0.89055920  
## [37] 0.89110407 0.89150530 0.89179271 0.89198805 0.89211952 0.89220514

**Inference:** The model using k-cross validation is extracted.

**6.Build the final model and interpret**

linRidgeMod = linearRidge(trainset$mpg ~ ., data = trainset)  
predicted = predict(linRidgeMod, testset) # predict on test data  
compare1 = cbind (actual=testset$mpg, predicted)  
mean (apply(compare1, 1, min)/apply(compare1, 1, max))

## [1] 0.9029484

summary(linRidgeMod)

##   
## Call:  
## linearRidge(formula = trainset$mpg ~ ., data = trainset)  
##   
##   
## Coefficients:  
## Estimate Scaled estimate Std. Error (scaled) t value (scaled)  
## (Intercept) 14.742140 NA NA NA  
## cyl -0.326153 -2.753488 3.321952 0.829  
## disp -0.005767 -3.492460 2.841631 1.229  
## hp -0.015200 -5.409057 3.161192 1.711  
## drat 1.757967 4.564728 3.400698 1.342  
## wt -1.322165 -6.502449 3.091123 2.104  
## qsec 0.468642 4.077506 3.255874 1.252  
## vs 0.103222 0.238042 3.398804 0.070  
## am 2.909703 6.710119 3.236733 2.073  
## gear -0.086758 -0.279297 3.248383 0.086  
## carb -0.277200 -2.201791 3.031816 0.726  
## Pr(>|t|)   
## (Intercept) NA   
## cyl 0.4072   
## disp 0.2191   
## hp 0.0871 .  
## drat 0.1795   
## wt 0.0354 \*  
## qsec 0.2104   
## vs 0.9442   
## am 0.0382 \*  
## gear 0.9315   
## carb 0.4677   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Ridge parameter: 0.1286921, chosen automatically, computed using 3 PCs  
##   
## Degrees of freedom: model 5.755 , variance 4.066 , residual 7.443

#Creating another model with only significant values.  
  
linRidgeMod = linearRidge(trainset$mpg ~ ., data = trainset[, c(6,10,11)])  
predicted1 = predict(linRidgeMod, testset) # predict on test data  
compare2 = cbind (actual=testset$mpg, predicted1)  
mean (apply(compare2, 1, min)/apply(compare2, 1, max))

## [1] 0.9464945

summary(linRidgeMod)

##   
## Call:  
## linearRidge(formula = trainset$mpg ~ ., data = trainset[, c(6,   
## 10, 11)])  
##   
##   
## Coefficients:  
## Estimate Scaled estimate Std. Error (scaled) t value (scaled)  
## (Intercept) 27.642 NA NA NA  
## wt -3.316 -16.308 4.659 3.500  
## gear 1.924 6.193 4.089 1.514  
## carb -1.395 -11.082 4.262 2.600  
## Pr(>|t|)   
## (Intercept) NA   
## wt 0.000465 \*\*\*  
## gear 0.129929   
## carb 0.009319 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Ridge parameter: 0.04208334, chosen automatically, computed using 2 PCs  
##   
## Degrees of freedom: model 2.698 , variance 2.457 , residual 2.939

**Inference:** The accuracy has increased from 75% to 88%.

RMSE = sqrt(mean((testset$mpg-predicted1)^2))  
RMSE

## [1] 1.389815

**Inference:** RMSE has decreased too.