Midterm

anton

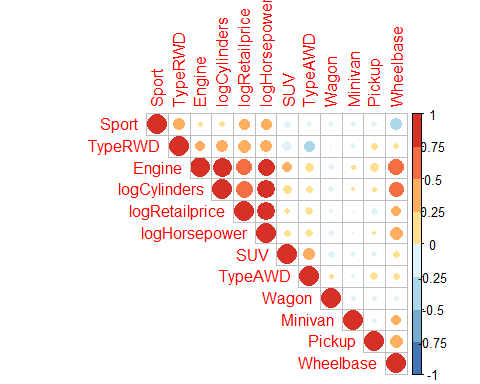
2023-03-05

library(dplyr)  
library(caret)  
library(glmnet)  
library(corrplot)  
library(RColorBrewer)  
library(car) # for VIF

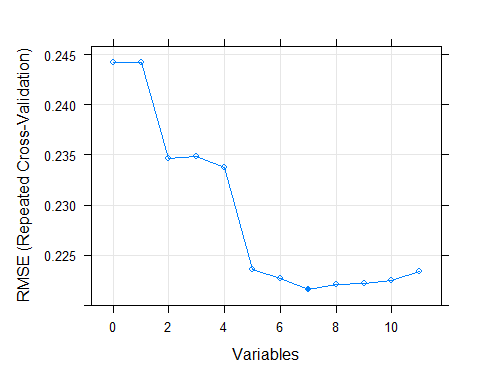
###########################################################  
## Clean up, transform, predictor selection  
  
myAuto = read.csv("04cars.csv")  
  
myAuto <- myAuto %>% select(-c(Height,Length))  
  
myAuto <- na.omit(myAuto)  
  
myAuto <- myAuto %>%  
 mutate(logRetailprice = log(Retailprice),  
 logCylinders = log(Cylinders),  
 TypeAWD = if\_else(Type == "AWD", 1, 0),  
 TypeRWD = if\_else(Type == "RWD", 1, 0),  
 logHorsepower = log(Horsepower)) %>%   
 select(-c(Type,Weight,CityMPG,HwyMPG,Retailprice,Cylinders,Horsepower))  
  
head(myAuto)

## Sport SUV Wagon Minivan Pickup Engine Wheelbase logRetailprice logCylinders  
## 1 0 0 0 0 0 1.6 95 9.237956 1.386294  
## 2 0 0 0 0 0 1.6 96 9.262838 1.386294  
## 3 0 0 0 0 0 1.5 93 9.283591 1.386294  
## 4 0 0 0 0 0 2.2 103 9.305196 1.386294  
## 5 0 0 0 0 0 1.6 95 9.319643 1.386294  
## 6 0 0 0 0 0 1.5 93 9.331673 1.386294  
## TypeAWD TypeRWD logHorsepower  
## 1 0 0 4.644391  
## 2 0 0 4.634729  
## 3 0 0 4.682131  
## 4 0 0 4.941642  
## 5 0 0 4.644391  
## 6 0 0 4.682131

###########################################################  
## Check correlation  
  
mycars\_numeric = select\_if(myAuto, is.numeric)  
mycars\_correlations <- cor(mycars\_numeric,   
 use = "pairwise.complete.obs")  
corrplot(mycars\_correlations,   
 type = "upper", order = "hclust",   
 col = rev(brewer.pal(n = 8, name = "RdYlBu")))



###########################################################  
## Recursive Feature Elimination  
  
set.seed(8)  
  
control\_rfe <- rfeControl(functions = lmFuncs,  
 method = "repeatedcv",  
 repeats = 5,  
 number = 5, #cv number  
 verbose = FALSE)  
  
Model.rfe = (logRetailprice ~ .)  
X\_train = model.matrix(Model.rfe, data = myAuto)[,-1]  
y\_train = myAuto$logRetailprice  
  
result\_rfe = rfe(x = X\_train,   
 y = y\_train,   
 sizes = c(1:dim(myAuto)[2]-1),  
 rfeControl = control\_rfe)  
  
plot(result\_rfe, type = c("g", "o"))



cbind(result\_rfe$results$Variables,result\_rfe$results$RMSE,result\_rfe$results$Rsquared)

## [,1] [,2] [,3]  
## [1,] 0 0.2442313 0.7511380  
## [2,] 1 0.2442313 0.7511380  
## [3,] 2 0.2346717 0.7704056  
## [4,] 3 0.2348867 0.7696778  
## [5,] 4 0.2337546 0.7712547  
## [6,] 5 0.2235883 0.7902189  
## [7,] 6 0.2226998 0.7918175  
## [8,] 7 0.2215720 0.7935745  
## [9,] 8 0.2220239 0.7928014  
## [10,] 9 0.2222139 0.7924825  
## [11,] 10 0.2224495 0.7920394  
## [12,] 11 0.2233403 0.7902819

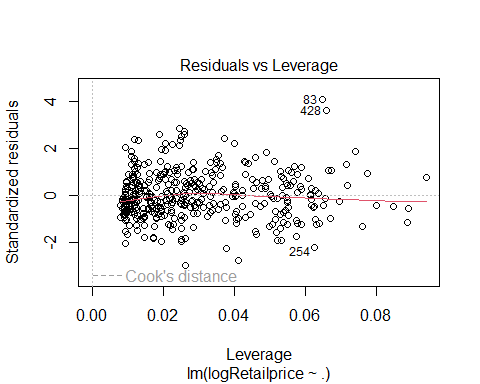
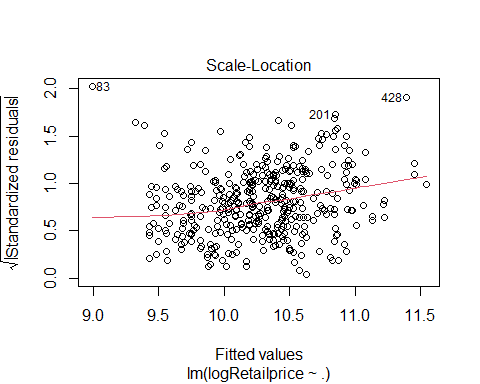
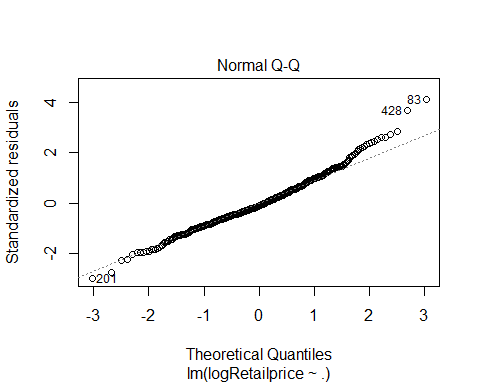
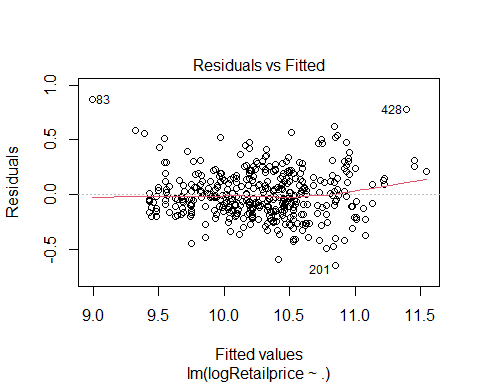
result\_rfe$optVariables

## [1] "logHorsepower" "Pickup" "logCylinders" "TypeAWD"   
## [5] "TypeRWD" "Sport" "SUV"

######################################################  
## Check on VIF  
  
testvif = lm(logRetailprice ~ ., data = myAuto)  
vif(testvif)

## Sport SUV Wagon Minivan Pickup   
## 2.004855 1.515528 1.074367 1.238674 1.752164   
## Engine Wheelbase logCylinders TypeAWD TypeRWD   
## 8.029331 3.650697 6.762143 1.534411 1.742702   
## logHorsepower   
## 4.058137

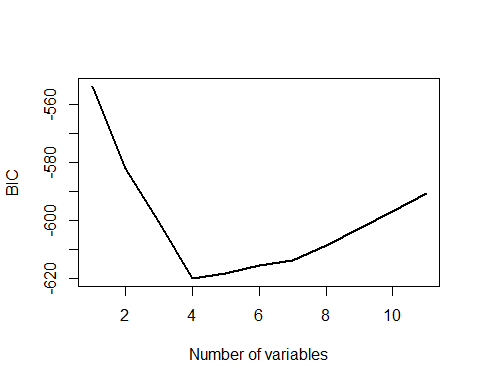
plot(testvif)



######################################################  
## Peeking on regsubset  
  
library(leaps)   
  
regfit.full = regsubsets(logRetailprice ~ ., data = myAuto,  
 method = "exhaustive", nvmax = dim(myAuto)-1)  
  
regfit.full.summary = summary(regfit.full)  
regfit.full.summary$bic

## [1] -553.8009 -582.1424 -600.3281 -619.8728 -618.2682 -615.3930 -613.8406  
## [8] -608.3963 -602.5874 -596.6846 -590.6698

plot(regfit.full.summary$bic, xlab = "Number of variables",  
 ylab = "BIC", type = "l", lwd = 2)



which.min(regfit.full.summary$bic)

## [1] 4

coef(regfit.full, which.min(regfit.full.summary$bic))

## (Intercept) Pickup TypeAWD TypeRWD logHorsepower   
## 3.9890288 -0.3928527 0.1542805 0.2066299 1.1687733

######################################################  
## Feature Selection using ENET Lasso, Alpha = 1  
library(doParallel)  
  
cls = makeCluster(detectCores()-1)  
registerDoParallel(cls)  
  
set.seed(8)  
  
#tuning params for ENET  
lambdalist = c((1:1000)/10000)  
  
n = dim(myAuto)[1]  
#n.alpha = length(alphalist)  
nfolds = 5  
  
groups = rep(1:nfolds,length=n)  
cvgroups = sample(groups,n)  
  
# store predicted values from the double-cross-validation  
allpredictedCV = rep(NA,n)  
allpredictedMethod = rep(NA,n)  
allpredictedMSE = rep(NA,n)  
  
# set up storage to see what models are "best" on the inner loops  
allbestRMSE = rep(NA,nfolds)  
allbestTypes = rep(NA,nfolds)  
allbestPars = vector("list",nfolds)  
  
Model.Full = (logRetailprice ~ .)  
Model.Penalized = (logRetailprice ~ .)  
  
timing <- system.time({  
for (j in 1:nfolds) { # loop through outer splits  
 groupj = (cvgroups == j)  
   
 # train data  
 traindata = myAuto[!groupj,]  
 trainx = model.matrix(Model.Full, data = traindata)[,-1]  
 trainy = traindata$logRetailprice  
   
 # test data  
 validdata = myAuto[groupj,]  
 validx = model.matrix(Model.Full, data = validdata)[,-1]  
 validy = validdata$logRetailprice  
   
 # all model-fitting process with traindata  
 dataused=traindata  
   
 training = trainControl(method = "cv", number = 5, allowParallel = TRUE)  
  
 # cross-validation of penalized regression  
 fit\_caret\_penalized = train(Model.Penalized,  
 data = dataused,  
 method = "glmnet",  
 trControl = training,  
 tuneGrid = expand.grid(alpha=1,lambda=lambdalist))  
   
 all\_best\_Types = c("ELASTICNET")  
 all\_best\_Pars = list(fit\_caret\_penalized$bestTune)  
 all\_best\_Models = list(glmnet(trainx, trainy, alpha=fit\_caret\_penalized$bestTune$alpha, lambda=lambdalist))  
 all\_best\_RMSE = c(min(fit\_caret\_penalized$results$RMSE))  
   
 # the best model  
 one\_best\_Type = all\_best\_Types[which.min(all\_best\_RMSE)]  
 one\_best\_Pars = all\_best\_Pars[which.min(all\_best\_RMSE)]  
 one\_best\_Model = all\_best\_Models[[which.min(all\_best\_RMSE)]]  
 one\_best\_RSME = all\_best\_RMSE[[which.min(all\_best\_RMSE)]]  
   
 # for checking later to see what's the best from each fold  
 allbestTypes[j] = one\_best\_Type  
 allbestPars[[j]] = one\_best\_Pars  
 allbestRMSE[j] = one\_best\_RSME  
   
 allpredictedMethod[groupj] = one\_best\_Type  
 allpredictedMSE[groupj] = one\_best\_RSME  
   
 if (one\_best\_Type == "ELASTICNET") {  
 ENETLAMBDA = one\_best\_Pars[[1]]$lambda  
 allpredictedCV[groupj] = predict(one\_best\_Model,newx=validx,s=ENETLAMBDA)  
 }  
}  
})  
  
paste("Elapsed times: ", timing["elapsed"])

## [1] "Elapsed times: 11.5"

stopCluster(cls)  
  
#see the best MSE  
one\_best\_RSME

## [1] 0.2210077

#see the best params  
paste("alpha",one\_best\_Pars[[1]]$alpha,"lambda",one\_best\_Pars[[1]]$lambda)

## [1] "alpha 1 lambda 0.0049"

#see the best Model  
bestcoef = coef(one\_best\_Model, s = one\_best\_Pars[[1]]$lambda)  
options(scipen=999)  
round(bestcoef,4)

## 12 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 4.1113  
## Sport 0.0843  
## SUV -0.0396  
## Wagon .   
## Minivan .   
## Pickup -0.3577  
## Engine .   
## Wheelbase .   
## logCylinders 0.1368  
## TypeAWD 0.1093  
## TypeRWD 0.1552  
## logHorsepower 1.1045

# print individually  
for (j in 1:nfolds) {  
 writemodel = paste("The best model at loop", j,   
 "is of type", allbestTypes[j],  
 "with parameter(s)",allbestPars[[j]])  
 print(writemodel, quote = FALSE)  
}

## [1] The best model at loop 1 is of type ELASTICNET with parameter(s) list(alpha = 1, lambda = 0.0066)  
## [1] The best model at loop 2 is of type ELASTICNET with parameter(s) list(alpha = 1, lambda = 0.0039)  
## [1] The best model at loop 3 is of type ELASTICNET with parameter(s) list(alpha = 1, lambda = 0.0039)  
## [1] The best model at loop 4 is of type ELASTICNET with parameter(s) list(alpha = 1, lambda = 0.0026)  
## [1] The best model at loop 5 is of type ELASTICNET with parameter(s) list(alpha = 1, lambda = 0.0049)

######################################################  
## DO NOT USE  
## RLM's predictors were selected from ENET  
##  
## Still figure out what Prof said, try to incorporate either of these:  
## a. If you're doing double CV, you can code it using 2 for loops instead of 1 for loop + caret, do the feature selection on the inner loop ## training set, and then use those features in the inner and outer loop test sets.  
## b. If you're using caret, you can use the Recursive Feature Elimination function (not something we'll get into in this course).  
##  
  
library(doParallel)  
  
cls = makeCluster(detectCores()-1)  
registerDoParallel(cls)  
  
set.seed(8)  
  
#tuning params for ENET  
lambdalist = c((1:1000)/10000)  
alphalist = seq(0, 1, by = 0.1)  
  
n = dim(myAuto)[1]  
n.alpha = length(alphalist)  
nfolds = 5  
  
groups = rep(1:nfolds,length=n)  
cvgroups = sample(groups,n)  
  
# store predicted values from the double-cross-validation  
allpredictedCV = rep(NA,n)  
allpredictedMethod = rep(NA,n)  
allpredictedMSE = rep(NA,n)  
  
# set up storage to see what models are "best" on the inner loops  
allbestRMSE = rep(NA,nfolds)  
allbestTypes = rep(NA,nfolds)  
allbestPars = vector("list",nfolds)  
  
Model.Full = (logRetailprice ~ .)  
Model.Penalized = (logRetailprice ~ .)  
Model.Robust = (logRetailprice ~ . -Wagon -Minivan -Engine -Wheelbase)  
  
timing <- system.time({  
for (j in 1:nfolds) { # loop through outer splits  
 groupj = (cvgroups == j)  
   
 # train data  
 traindata = myAuto[!groupj,]  
 trainx = model.matrix(Model.Full, data = traindata)[,-1]  
 trainy = traindata$logRetailprice  
   
 # test data  
 validdata = myAuto[groupj,]  
 validx = model.matrix(Model.Full, data = validdata)[,-1]  
 validy = validdata$logRetailprice  
   
 # all model-fitting process with traindata  
 dataused=traindata  
   
 training = trainControl(method = "cv", number = 5, allowParallel = TRUE)  
  
 # cross-validation of penalized regression  
 fit\_caret\_penalized = train(Model.Penalized,  
 data = dataused,  
 method = "glmnet",  
 trControl = training,  
 tuneGrid = expand.grid(alpha=alphalist,lambda=lambdalist))  
   
 # cross-validation of robust regression   
 fit\_caret\_robust = train(Model.Robust,  
 data = dataused,  
 method = "rlm",  
 trControl = training)  
   
 # all best models  
 all\_best\_Types = c("ELASTICNET", "ROBUST")  
 all\_best\_Pars = list(fit\_caret\_penalized$bestTune, fit\_caret\_robust$bestTune)  
 all\_best\_Models = list(glmnet(trainx, trainy, alpha=fit\_caret\_penalized$bestTune$alpha, lambda=lambdalist),  
 fit\_caret\_robust$finalModel)  
 all\_best\_RMSE = c(min(fit\_caret\_penalized$results$RMSE),min(fit\_caret\_robust$results$RMSE))  
   
 # the best model  
 one\_best\_Type = all\_best\_Types[which.min(all\_best\_RMSE)]  
 one\_best\_Pars = all\_best\_Pars[which.min(all\_best\_RMSE)]  
 one\_best\_Model = all\_best\_Models[[which.min(all\_best\_RMSE)]]  
 one\_best\_RSME = all\_best\_RMSE[[which.min(all\_best\_RMSE)]]  
   
 # for checking later to see what's the best from each fold  
 allbestTypes[j] = one\_best\_Type  
 allbestPars[[j]] = one\_best\_Pars  
 allbestRMSE[j] = one\_best\_RSME  
   
 allpredictedMethod[groupj] = one\_best\_Type  
 allpredictedMSE[groupj] = one\_best\_RSME  
   
 if (one\_best\_Type == "ELASTICNET") {  
 ENETLAMBDA = one\_best\_Pars[[1]]$lambda  
 allpredictedCV[groupj] = predict(one\_best\_Model,newx=validx,s=ENETLAMBDA)  
 } else if (one\_best\_Type == "ROBUST") {  
 allpredictedCV[groupj] = predict(one\_best\_Model,newdata=validdata)  
 }  
}  
})  
  
paste("Elapsed times: ", timing["elapsed"])

## [1] "Elapsed times: 88.33"

stopCluster(cls)  
  
#see the best model  
one\_best\_Type

## [1] "ROBUST"

#see the best pars  
#one\_best\_Pars[[1]]$lambda  
#one\_best\_Pars[[1]]$alpha  
#print(one\_best\_Pars, quote = FALSE)  
print(one\_best\_Pars[[1]], quote = FALSE)

## intercept psi  
## 6 TRUE psi.bisquare

paste("alpha",one\_best\_Pars[[1]]$alpha,"lambda",one\_best\_Pars[[1]]$lambda)

## [1] "alpha lambda "

#see the best MSE  
one\_best\_RSME

## [1] 0.218316

#see the best Model  
bestcoef = coef(one\_best\_Model, s = one\_best\_Pars[[1]]$lambda)  
options(scipen=999)  
round(bestcoef,4)

## (Intercept) Sport SUV Pickup logCylinders   
## 4.1611 0.1096 -0.0761 -0.3951 0.1841   
## TypeAWD TypeRWD logHorsepower   
## 0.1486 0.1762 1.0770

#see all predicted CV  
#allpredicted <- cbind(allpredictedMethod,allpredictedCV,allpredictedMSE)  
#allpredicted  
#allpredicted[allpredictedMethod == "ELASTICNET", ]  
  
### Outer model assessments  
y = myAuto$logRetailprice  
  
# get CV assessment  
CV.assess = mean((allpredictedCV-y)^2)  
CV.assess

## [1] 0.04852664

# get RMSE assessment  
RMSE = sqrt(mean((allpredictedCV-y)^2)); RMSE

## [1] 0.2202876

# get R2 assessment  
R2 = 1-sum((allpredictedCV-y)^2)/sum((y-mean(y))^2); R2

## [1] 0.7928617

# print individually  
for (j in 1:nfolds) {  
 writemodel = paste("The best model at loop", j,   
 "is of type", allbestTypes[j],  
 "with parameter(s)",allbestPars[[j]])  
 print(writemodel, quote = FALSE)  
}

## [1] The best model at loop 1 is of type ROBUST with parameter(s) list(intercept = TRUE, psi = 2)  
## [1] The best model at loop 2 is of type ROBUST with parameter(s) list(intercept = TRUE, psi = 2)  
## [1] The best model at loop 3 is of type ROBUST with parameter(s) list(intercept = TRUE, psi = 2)  
## [1] The best model at loop 4 is of type ROBUST with parameter(s) list(intercept = TRUE, psi = 2)  
## [1] The best model at loop 5 is of type ROBUST with parameter(s) list(intercept = TRUE, psi = 3)