library(ElemStatLearn) #contains the data

library(car) #package to calculate Variance Inflation Factor

library(corrplot) #correlation plots

library(leaps) #best subsets regression

library(glmnet) #allows ridge regression, LASSO and elastic net

library(caret) #this will help identify the appropriate parameters

data(prostate)

str(prostate)

plot(prostate)

plot(prostate$gleason)

table(prostate$gleason)

boxplot(prostate$lpsa~prostate$gleason, xlab="Gleason Score", ylab="Log of PSA")

prostate$gleason = ifelse(prostate$gleason == 6, 0, 1)

table(prostate$gleason)

p.cor = cor(prostate)

corrplot.mixed(p.cor)

train = subset(prostate, train==TRUE)[,1:9]

str(train)

test = subset(prostate, train==FALSE)[,1:9]

str(test)

subfit = regsubsets(lpsa~., data=train)

b.sum = summary(subfit)

which.min(b.sum$bic)

plot(b.sum$bic, type="l", xlab="# of Features", ylab="BIC", main="BIC score by Feature Inclusion")

plot(subfit, scale="bic", main="Best Subset Features")

ols = lm(lpsa~lcavol+lweight+gleason, data=train)

plot(ols$fitted.values, train$lpsa, xlab="Predicted", ylab="Actual", main="Predicted vs Actual")

###

pred.subfit = predict(ols, newdata=test)

plot(pred.subfit, test$lpsa , xlab="Predicted", ylab="Actual", main="Predicted vs Actual")

resid.subfit = test$lpsa - pred.subfit

mean(resid.subfit^2)

x = as.matrix(train[,1:8])

y = train[ ,9]

ridge = glmnet(x, y, family="gaussian", alpha=0)

print(ridge)

plot(ridge, label=TRUE)

plot(ridge, xvar="lambda", label=TRUE)

ridge.coef = coef(ridge, s=0.1, exact=TRUE)

ridge.coef

plot(ridge, xvar="dev", label=TRUE)

newx = as.matrix(test[,1:8])

ridge.y = predict(ridge, newx=newx, type="response", s=0.1)

plot(ridge.y, test$lpsa, xlab="Predicted", ylab="Actual", main="Ridge Regression")

ridge.resid = ridge.y - test$lpsa

mean(ridge.resid^2)

lasso = glmnet(x, y, family="gaussian", alpha=1)

print(lasso)

plot(lasso, xvar="lambda", label=TRUE)

lasso.coef = coef(lasso, s=0.045, exact=TRUE)

lasso.coef

lasso.y = predict(lasso, newx=newx, type="response", s=0.045)

plot(lasso.y, test$lpsa, xlab="Predicted", ylab="Actual", main="LASSO")

lasso.resid = lasso.y - test$lpsa

mean(lasso.resid^2)

grid = expand.grid(.alpha=seq(0,1, by=.2), .lambda=seq(0.00,0.2, by=0.02))

table(grid)

head(grid)

control = trainControl(method="LOOCV") #selectionFunction="best"

set.seed(701) #our random seed

enet.train = train(lpsa~., data=train, method="glmnet", trControl=control, tuneGrid=grid)

enet.train

enet = glmnet(x, y,family="gaussian", alpha=0, lambda=.08)

enet.coef = coef(enet, s=.08, exact=TRUE)

enet.coef

enet.y = predict(enet, newx=newx, type="response", s=.08)

plot(enet.y, test$lpsa, xlab="Predicted", ylab="Actual", main="Elastic Net")

enet.resid = enet.y - test$lpsa

mean(enet.resid^2)

set.seed(317)

lasso.cv = cv.glmnet(x, y)

plot(lasso.cv)

lasso.cv$lambda.min #minimum

lasso.cv$lambda.1se #one standard error away

coef(lasso.cv, s ="lambda.1se")

lasso.y.cv = predict(lasso.cv, newx=newx, type="response", s="lambda.1se")

lasso.cv.resid = lasso.y.cv - test$lpsa

mean(lasso.cv.resid^2)