Homework 1

Preston Robertson

Setember 24 2021

#### 1. Fit a linear regression model using Sepal.Length as response and Sepal.Width as predictor

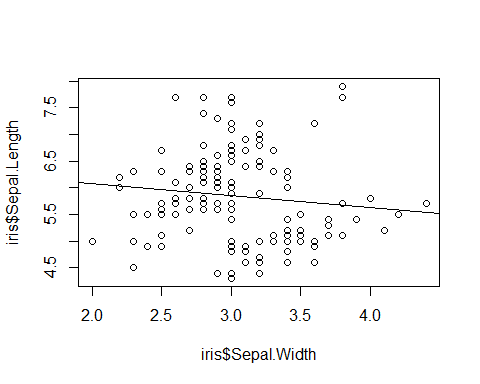
#### 1.A Fit the regression model

library(datasets)  
data(iris)  
  
lmfit.Sepal<-lm(formula=Sepal.Length ~ Sepal.Width, data=iris)  
summary(lmfit.Sepal)

##   
## Call:  
## lm(formula = Sepal.Length ~ Sepal.Width, data = iris)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.5561 -0.6333 -0.1120 0.5579 2.2226   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 6.5262 0.4789 13.63 <2e-16 \*\*\*  
## Sepal.Width -0.2234 0.1551 -1.44 0.152   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8251 on 148 degrees of freedom  
## Multiple R-squared: 0.01382, Adjusted R-squared: 0.007159   
## F-statistic: 2.074 on 1 and 148 DF, p-value: 0.1519

#### 1.A.B Plot the data with the fitted regression line

par(mfcol=c(1,1))  
plot(iris$Sepal.Width, iris$Sepal.Length)  
abline(lmfit.Sepal)



#### 1.B. Based on your answer in 1c, fit a new model (Model 1) using Sepal.Length as response, Sepal.Width and Species as predictors.

lmfit.Sepal<-lm(formula=Sepal.Length ~ Sepal.Width+Species, data=iris)  
summary(lmfit.Sepal)

##   
## Call:  
## lm(formula = Sepal.Length ~ Sepal.Width + Species, data = iris)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.30711 -0.25713 -0.05325 0.19542 1.41253   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.2514 0.3698 6.089 9.57e-09 \*\*\*  
## Sepal.Width 0.8036 0.1063 7.557 4.19e-12 \*\*\*  
## Speciesversicolor 1.4587 0.1121 13.012 < 2e-16 \*\*\*  
## Speciesvirginica 1.9468 0.1000 19.465 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.438 on 146 degrees of freedom  
## Multiple R-squared: 0.7259, Adjusted R-squared: 0.7203   
## F-statistic: 128.9 on 3 and 146 DF, p-value: < 2.2e-16

#### 1.C. Can you come up with another model that have better adj R2 than Model 1

lmfit.Sepal<-lm(formula=Sepal.Length ~ Sepal.Width+Petal.Width+Petal.Length+Species, data=iris)  
summary(lmfit.Sepal)

##   
## Call:  
## lm(formula = Sepal.Length ~ Sepal.Width + Petal.Width + Petal.Length +   
## Species, data = iris)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.79424 -0.21874 0.00899 0.20255 0.73103   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.17127 0.27979 7.760 1.43e-12 \*\*\*  
## Sepal.Width 0.49589 0.08607 5.761 4.87e-08 \*\*\*  
## Petal.Width -0.31516 0.15120 -2.084 0.03889 \*   
## Petal.Length 0.82924 0.06853 12.101 < 2e-16 \*\*\*  
## Speciesversicolor -0.72356 0.24017 -3.013 0.00306 \*\*   
## Speciesvirginica -1.02350 0.33373 -3.067 0.00258 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3068 on 144 degrees of freedom  
## Multiple R-squared: 0.8673, Adjusted R-squared: 0.8627   
## F-statistic: 188.3 on 5 and 144 DF, p-value: < 2.2e-16

# NOTE  
# This is the best rsq that I personally found. I did not include my other attempts because my computer is slow and takes a long time to run this homework file.

#### 1.D. Use subset selection method (step) to explore if we can further improve the fit by incorporating more interaction terms? This will be your Model 3.

#FORWARD  
  
lm.null<-lm(Sepal.Length~ 1, data=iris)  
lm.aic.forward<-step(lm.null,direction="forward",trace=1,scope= ~ Sepal.Width\*Species+Petal.Width\*Species+Petal.Length\*Species+Petal.Width\*Petal.Length+Sepal.Width\*Petal.Length+Petal.Width\*Sepal.Width)

## Start: AIC=-55.6  
## Sepal.Length ~ 1  
##   
## Df Sum of Sq RSS AIC  
## + Petal.Length 1 77.643 24.525 -267.641  
## + Petal.Width 1 68.353 33.815 -219.460  
## + Species 2 63.212 38.956 -196.230  
## + Sepal.Width 1 1.412 100.756 -55.690  
## <none> 102.168 -55.602  
##   
## Step: AIC=-267.64  
## Sepal.Length ~ Petal.Length  
##   
## Df Sum of Sq RSS AIC  
## + Sepal.Width 1 8.1963 16.329 -326.66  
## + Species 2 7.8434 16.682 -321.45  
## + Petal.Width 1 0.6443 23.881 -269.63  
## <none> 24.525 -267.64  
##   
## Step: AIC=-326.66  
## Sepal.Length ~ Petal.Length + Sepal.Width  
##   
## Df Sum of Sq RSS AIC  
## + Species 2 2.36325 13.966 -346.11  
## + Petal.Width 1 1.88336 14.445 -343.04  
## + Sepal.Width:Petal.Length 1 0.35021 15.979 -327.91  
## <none> 16.329 -326.66  
##   
## Step: AIC=-346.11  
## Sepal.Length ~ Petal.Length + Sepal.Width + Species  
##   
## Df Sum of Sq RSS AIC  
## + Sepal.Width:Petal.Length 1 0.42066 13.545 -348.69  
## + Petal.Width 1 0.40903 13.556 -348.57  
## + Species:Petal.Length 2 0.58174 13.384 -348.49  
## + Sepal.Width:Species 2 0.56406 13.402 -348.29  
## <none> 13.966 -346.11  
##   
## Step: AIC=-348.69  
## Sepal.Length ~ Petal.Length + Sepal.Width + Species + Petal.Length:Sepal.Width  
##   
## Df Sum of Sq RSS AIC  
## + Species:Petal.Length 2 0.79054 12.754 -353.71  
## + Petal.Width 1 0.26380 13.281 -349.64  
## <none> 13.545 -348.69  
## + Sepal.Width:Species 2 0.30033 13.245 -348.06  
##   
## Step: AIC=-353.71  
## Sepal.Length ~ Petal.Length + Sepal.Width + Species + Petal.Length:Sepal.Width +   
## Petal.Length:Species  
##   
## Df Sum of Sq RSS AIC  
## <none> 12.754 -353.71  
## + Petal.Width 1 0.143170 12.611 -353.41  
## + Sepal.Width:Species 2 0.067286 12.687 -350.51

summary(lm.aic.forward)

##   
## Call:  
## lm(formula = Sepal.Length ~ Petal.Length + Sepal.Width + Species +   
## Petal.Length:Sepal.Width + Petal.Length:Species, data = iris)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.78426 -0.20523 0.00137 0.19134 0.69631   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.81642 0.59861 3.034 0.00287 \*\*   
## Petal.Length 0.65478 0.27106 2.416 0.01698 \*   
## Sepal.Width 0.80547 0.15577 5.171 7.78e-07 \*\*\*  
## Speciesversicolor -1.00414 0.54966 -1.827 0.06983 .   
## Speciesvirginica -2.84319 0.57878 -4.912 2.44e-06 \*\*\*  
## Petal.Length:Sepal.Width -0.10528 0.03977 -2.647 0.00903 \*\*   
## Petal.Length:Speciesversicolor 0.31787 0.26301 1.209 0.22884   
## Petal.Length:Speciesvirginica 0.59974 0.26037 2.303 0.02271 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2997 on 142 degrees of freedom  
## Multiple R-squared: 0.8752, Adjusted R-squared: 0.869   
## F-statistic: 142.2 on 7 and 142 DF, p-value: < 2.2e-16

#BACKWARDS  
  
lm.null<-lm(Sepal.Length~ 1, data=iris)  
lm.aic.backward<-step(lm.null,direction="backward",trace=1,scope= ~ Sepal.Width\*Species+Petal.Width\*Species+Petal.Length\*Species+Petal.Width\*Petal.Length+Sepal.Width\*Petal.Length+Petal.Width\*Sepal.Width)

## Start: AIC=-55.6  
## Sepal.Length ~ 1

summary(lm.aic.backward)

##   
## Call:  
## lm(formula = Sepal.Length ~ 1, data = iris)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.54333 -0.74333 -0.04333 0.55667 2.05667   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 5.84333 0.06761 86.42 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.8281 on 149 degrees of freedom

#BOTH  
  
lm.null<-lm(Sepal.Length~ 1, data=iris)  
lm.aic.both<-step(lm.null,direction="both",trace=1,scope= ~ Sepal.Width\*Species+Petal.Width\*Species+Petal.Length\*Species+Petal.Width\*Petal.Length+Sepal.Width\*Petal.Length+Petal.Width\*Sepal.Width)

## Start: AIC=-55.6  
## Sepal.Length ~ 1  
##   
## Df Sum of Sq RSS AIC  
## + Petal.Length 1 77.643 24.525 -267.641  
## + Petal.Width 1 68.353 33.815 -219.460  
## + Species 2 63.212 38.956 -196.230  
## + Sepal.Width 1 1.412 100.756 -55.690  
## <none> 102.168 -55.602  
##   
## Step: AIC=-267.64  
## Sepal.Length ~ Petal.Length  
##   
## Df Sum of Sq RSS AIC  
## + Sepal.Width 1 8.196 16.329 -326.66  
## + Species 2 7.843 16.682 -321.45  
## + Petal.Width 1 0.644 23.881 -269.63  
## <none> 24.525 -267.64  
## - Petal.Length 1 77.643 102.168 -55.60  
##   
## Step: AIC=-326.66  
## Sepal.Length ~ Petal.Length + Sepal.Width  
##   
## Df Sum of Sq RSS AIC  
## + Species 2 2.363 13.966 -346.11  
## + Petal.Width 1 1.883 14.445 -343.04  
## + Sepal.Width:Petal.Length 1 0.350 15.979 -327.91  
## <none> 16.329 -326.66  
## - Sepal.Width 1 8.196 24.525 -267.64  
## - Petal.Length 1 84.427 100.756 -55.69  
##   
## Step: AIC=-346.11  
## Sepal.Length ~ Petal.Length + Sepal.Width + Species  
##   
## Df Sum of Sq RSS AIC  
## + Sepal.Width:Petal.Length 1 0.4207 13.545 -348.69  
## + Petal.Width 1 0.4090 13.556 -348.57  
## + Species:Petal.Length 2 0.5817 13.384 -348.49  
## + Sepal.Width:Species 2 0.5641 13.402 -348.29  
## <none> 13.966 -346.11  
## - Species 2 2.3632 16.329 -326.66  
## - Sepal.Width 1 2.7161 16.682 -321.45  
## - Petal.Length 1 14.0382 28.004 -243.74  
##   
## Step: AIC=-348.69  
## Sepal.Length ~ Petal.Length + Sepal.Width + Species + Petal.Length:Sepal.Width  
##   
## Df Sum of Sq RSS AIC  
## + Species:Petal.Length 2 0.79054 12.754 -353.71  
## + Petal.Width 1 0.26380 13.281 -349.64  
## <none> 13.545 -348.69  
## + Sepal.Width:Species 2 0.30033 13.245 -348.06  
## - Petal.Length:Sepal.Width 1 0.42066 13.966 -346.11  
## - Species 2 2.43369 15.979 -327.91  
##   
## Step: AIC=-353.71  
## Sepal.Length ~ Petal.Length + Sepal.Width + Species + Petal.Length:Sepal.Width +   
## Petal.Length:Species  
##   
## Df Sum of Sq RSS AIC  
## <none> 12.754 -353.71  
## + Petal.Width 1 0.14317 12.611 -353.41  
## + Sepal.Width:Species 2 0.06729 12.687 -350.51  
## - Petal.Length:Species 2 0.79054 13.545 -348.69  
## - Petal.Length:Sepal.Width 1 0.62946 13.384 -348.49

summary(lm.aic.both)

##   
## Call:  
## lm(formula = Sepal.Length ~ Petal.Length + Sepal.Width + Species +   
## Petal.Length:Sepal.Width + Petal.Length:Species, data = iris)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.78426 -0.20523 0.00137 0.19134 0.69631   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.81642 0.59861 3.034 0.00287 \*\*   
## Petal.Length 0.65478 0.27106 2.416 0.01698 \*   
## Sepal.Width 0.80547 0.15577 5.171 7.78e-07 \*\*\*  
## Speciesversicolor -1.00414 0.54966 -1.827 0.06983 .   
## Speciesvirginica -2.84319 0.57878 -4.912 2.44e-06 \*\*\*  
## Petal.Length:Sepal.Width -0.10528 0.03977 -2.647 0.00903 \*\*   
## Petal.Length:Speciesversicolor 0.31787 0.26301 1.209 0.22884   
## Petal.Length:Speciesvirginica 0.59974 0.26037 2.303 0.02271 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2997 on 142 degrees of freedom  
## Multiple R-squared: 0.8752, Adjusted R-squared: 0.869   
## F-statistic: 142.2 on 7 and 142 DF, p-value: < 2.2e-16

#RESULTS  
#Running the data with the maximum amount of interactions increases the RSQ and decreases the RSE of the model increasing the accuracy of the forward/backward step method. This makes sense since the model has more interactions to test.

#### Question 2: Car Data

#### Cars data set: Elastic Net

# Load libraries, get data & set seed for reproducibility ---------------------  
set.seed(123)   
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-2

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(psych)   
  
data("mtcars")  
  
y <- mtcars %>% select(mpg) %>% scale(center = TRUE, scale = FALSE) %>% as.matrix()  
X <- mtcars %>% select(-mpg) %>% as.matrix()

#### 2.A Use the same data set we used in class in Lab 3. Explore how the performance of an Elastic net model varies w.r.t,Different lambda, Different alpha

#Changing Lamda  
  
alpha <- 1   
lambda <- .25  
base\_model\_cv <- glmnet(X, y, alpha = 1 , lambda = .25, standardize = TRUE)  
base\_y\_hat\_cv <- predict(base\_model\_cv, X)  
base\_ssr\_cv <- t(y - base\_y\_hat\_cv) %\*% (y - base\_y\_hat\_cv)  
base\_rsq\_cv <- cor(y, base\_y\_hat\_cv)^2  
  
sprintf('Alpha = %f', alpha)

## [1] "Alpha = 1.000000"

sprintf('Lambda = %f', lambda)

## [1] "Lambda = 0.250000"

sprintf('RSQ = %f', base\_rsq\_cv)

## [1] "RSQ = 0.858719"

sprintf('SSR = %f', base\_ssr\_cv)

## [1] "SSR = 161.986968"

#Changing Lamda  
  
alpha <- 1   
lambda <- .75  
base\_model\_cv <- glmnet(X, y, alpha = 1 , lambda = .75, standardize = TRUE)  
base\_y\_hat\_cv <- predict(base\_model\_cv, X)  
base\_ssr\_cv <- t(y - base\_y\_hat\_cv) %\*% (y - base\_y\_hat\_cv)  
base\_rsq\_cv <- cor(y, base\_y\_hat\_cv)^2  
  
sprintf('Alpha = %f', alpha)

## [1] "Alpha = 1.000000"

sprintf('Lambda = %f', lambda)

## [1] "Lambda = 0.750000"

sprintf('RSQ = %f', base\_rsq\_cv)

## [1] "RSQ = 0.842547"

sprintf('SSR = %f', base\_ssr\_cv)

## [1] "SSR = 198.391576"

#Results as Lambda increases the RSQ decreases but SSR increases.

#Changing Alpha  
  
alpha <- .5  
lambda <- .75  
base\_model\_cv <- glmnet(X, y, alpha = .5 , lambda = .75, standardize = TRUE)  
base\_y\_hat\_cv <- predict(base\_model\_cv, X)  
base\_ssr\_cv <- t(y - base\_y\_hat\_cv) %\*% (y - base\_y\_hat\_cv)  
base\_rsq\_cv <- cor(y, base\_y\_hat\_cv)^2  
  
sprintf('Alpha = %f', alpha)

## [1] "Alpha = 0.500000"

sprintf('Lambda = %f', lambda)

## [1] "Lambda = 0.750000"

sprintf('RSQ = %f', base\_rsq\_cv)

## [1] "RSQ = 0.855558"

sprintf('SSR = %f', base\_ssr\_cv)

## [1] "SSR = 171.851064"

#Changing Alpha  
  
alpha <- 0   
lambda <- .75  
base\_model\_cv <- glmnet(X, y, alpha = 0, lambda = .75, standardize = TRUE)  
base\_y\_hat\_cv <- predict(base\_model\_cv, X)  
base\_ssr\_cv <- t(y - base\_y\_hat\_cv) %\*% (y - base\_y\_hat\_cv)  
base\_rsq\_cv <- cor(y, base\_y\_hat\_cv)^2  
  
sprintf('Alpha = %f', alpha)

## [1] "Alpha = 0.000000"

sprintf('Lambda = %f', lambda)

## [1] "Lambda = 0.750000"

sprintf('RSQ = %f', base\_rsq\_cv)

## [1] "RSQ = 0.860291"

sprintf('SSR = %f', base\_ssr\_cv)

## [1] "SSR = 158.107764"

#RESULTS  
#If alpha increases then RSQ decreases and the SSR increases.

#### 2.B Based on your results in 1, can you find a better model than ridge and the LASSO? What is the evaluation criteria you use?

# Ridge Regression  
lambdas\_to\_try <- 10^seq(-3, 5, length.out = 100)  
ridge\_cv <- cv.glmnet(X, y, alpha = 0, lambda = lambdas\_to\_try,  
 standardize = TRUE, nfolds = 10)  
  
lambda\_cv <- ridge\_cv$lambda.min  
  
model\_cv <- glmnet(X, y, alpha = 0, lambda = lambda\_cv, standardize = TRUE)  
y\_hat\_cv <- predict(model\_cv, X)  
ssr\_cv <- t(y - y\_hat\_cv) %\*% (y - y\_hat\_cv)  
rsq\_ridge\_cv <- cor(y, y\_hat\_cv)^2

# Ridge Regression + AIC/BIC -----------------------------------  
X\_scaled <- scale(X)  
aic <- c()  
bic <- c()  
for (i in seq(lambdas\_to\_try)) {  
 model <- glmnet(X, y, alpha = 0, lambda = lambdas\_to\_try[i], standardize = TRUE)  
 betas <- as.vector((as.matrix(coef(model))[-1, ]))  
 resid <- y - (X\_scaled %\*% betas)  
 ld <- lambdas\_to\_try[i] \* diag(ncol(X\_scaled))  
 H <- X\_scaled %\*% solve(t(X\_scaled) %\*% X\_scaled + ld) %\*% t(X\_scaled)  
 df <- tr(H)  
 aic[i] <- nrow(X\_scaled) \* log(t(resid) %\*% resid) + 2 \* df  
 bic[i] <- nrow(X\_scaled) \* log(t(resid) %\*% resid) + 2 \* df \* log(nrow(X\_scaled))  
}  
  
  
lambda\_aic <- lambdas\_to\_try[which.min(aic)]  
lambda\_bic <- lambdas\_to\_try[which.min(bic)]  
  
model\_aic <- glmnet(X, y, alpha = 0, lambda = lambda\_aic, standardize = TRUE)  
y\_hat\_aic <- predict(model\_aic, X)  
ssr\_aic <- t(y - y\_hat\_aic) %\*% (y - y\_hat\_aic)  
rsq\_ridge\_aic <- cor(y, y\_hat\_aic)^2  
  
model\_bic <- glmnet(X, y, alpha = 0, lambda = lambda\_bic, standardize = TRUE)  
y\_hat\_bic <- predict(model\_bic, X)  
ssr\_bic <- t(y - y\_hat\_bic) %\*% (y - y\_hat\_bic)  
rsq\_ridge\_bic <- cor(y, y\_hat\_bic)^2

# LASSO  
lambdas\_to\_try <- 10^seq(-3, 5, length.out = 100)  
lasso\_cv <- cv.glmnet(X, y, alpha = 1, lambda = lambdas\_to\_try,  
 standardize = TRUE, nfolds = 10)  
  
lambda\_cv <- lasso\_cv$lambda.min  
model\_cv <- glmnet(X, y, alpha = 1, lambda = lambda\_cv, standardize = TRUE)  
model\_cv

##   
## Call: glmnet(x = X, y = y, alpha = 1, lambda = lambda\_cv, standardize = TRUE)   
##   
## Df %Dev Lambda  
## 1 3 82.05 0.8111

y\_hat\_cv <- predict(model\_cv, X)  
ssr\_cv <- t(y - y\_hat\_cv) %\*% (y - y\_hat\_cv)  
rsq\_lasso\_cv <- cor(y, y\_hat\_cv)^2

# Elastic Net  
set.seed(11)  
lambdas\_to\_try <- 10^seq(-3, 5, length.out = 100)  
alpha\_to\_try<- seq(0.0, 0.99, length.out = 10)  
df <- data.frame(matrix(ncol=3))  
  
for(i in 1:length(alpha\_to\_try<1)){  
 a = alpha\_to\_try[i]  
   
 ElasticNet\_cv <- cv.glmnet(X, y, alpha = a, lambda = lambdas\_to\_try,  
 standardize = TRUE)  
l = ElasticNet\_cv$lambda.min  
  
Model\_cv <- glmnet(X, y, alpha = a, lambda = 1, standardize = TRUE)  
  
y\_hat\_cv <- predict(Model\_cv, X)  
ssr\_cv <- t(y - y\_hat\_cv) %\*% (y - y\_hat\_cv)  
r = rsq\_ElasticNet\_cv <- cor(y, y\_hat\_cv)^2  
  
d = c(a,l,rsq\_ElasticNet\_cv)  
df <- rbind(df,d)}  
  
sprintf('Alpha = %f', a)

## [1] "Alpha = 0.990000"

sprintf('Lambda = %f', lambda\_cv)

## [1] "Lambda = 0.811131"

sprintf('RSQ = %f', rsq\_ElasticNet\_cv)

## [1] "RSQ = 0.842011"

sprintf('SSR = %f', ssr\_cv)

## [1] "SSR = 214.758906"

print(df)

## X1 X2 X3  
## 1 NA NA NA  
## 2 0.00 2.4770764 0.8589281  
## 3 0.11 2.0565123 0.8583733  
## 4 0.22 1.4174742 0.8570265  
## 5 0.33 1.1768120 0.8550163  
## 6 0.44 0.6734151 0.8541662  
## 7 0.55 0.8111308 0.8522767  
## 8 0.66 0.8111308 0.8494200  
## 9 0.77 0.8111308 0.8443311  
## 10 0.88 0.6734151 0.8425497  
## 11 0.99 0.6734151 0.8420108

#RESULTS  
#Based on these results, we can see that an Alpha = 0 and Lambda = 2.47 will yield the best RSQ.Which means that our model works best with a Ridge Regression.

rsq <- cbind("R-squared" = c(rsq\_ridge\_cv, rsq\_ridge\_aic, rsq\_ridge\_bic, rsq\_lasso\_cv, rsq\_ElasticNet\_cv))  
rownames(rsq) <- c("ridge cross-validated", "ridge AIC", "ridge BIC", "lasso cross\_validated", "ElasticNet\_CV")  
print(rsq)

## R-squared  
## ridge cross-validated 0.8524086  
## ridge AIC 0.8496310  
## ridge BIC 0.8412011  
## lasso cross\_validated 0.8424217  
## ElasticNet\_CV 0.8420108

#RESULTS  
#Based on these results the best model from the selected is Elastic Net. Which is better than Ridge and LASSO individually.

#### Q3: Cars data set: PCR and PLS

#### Use the cars data set, and use “mpg” as the response variables. Use 75% of the samples to train and validation the data, and leave 25% of the samples for testing.

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.4 v stringr 1.4.0  
## v tidyr 1.1.3 v forcats 0.5.1  
## v readr 2.0.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x ggplot2::%+%() masks psych::%+%()  
## x ggplot2::alpha() masks psych::alpha()  
## 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()

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(pls)

##   
## Attaching package: 'pls'

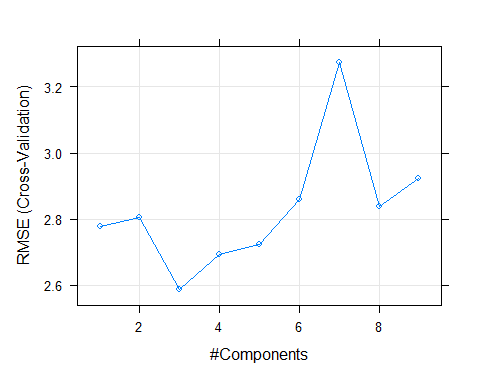
## The following object is masked from 'package:caret':  
##   
## R2

## The following object is masked from 'package:stats':  
##   
## loadings

data("Boston", package = "MASS")

#### 3.1 Apply the PCR and PLS, respectively. What is the best number of components extracted for PCR and PLS, respectively?

set.seed(123)  
training.samples <- mtcars$mpg %>%  
 createDataPartition(p = 0.75, list = FALSE)  
train.data <- mtcars[training.samples, ]  
test.data <- mtcars[-training.samples, ]  
  
  
pcr\_model <- train(  
 mpg~., data = train.data, method = "pcr",  
 scale = TRUE,  
 trControl = trainControl("cv", number = 10),  
 tuneLength = 10  
 )  
plot(pcr\_model)



pcr\_model$bestTune

## ncomp  
## 3 3

summary(pcr\_model$finalModel)

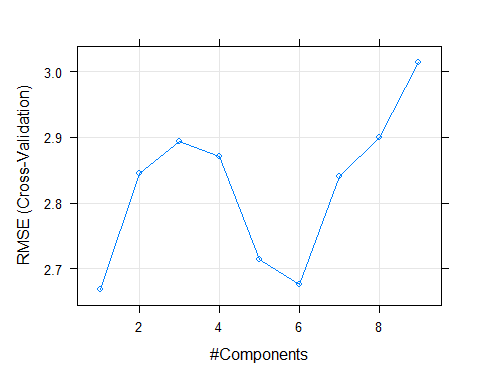
## Data: X dimension: 25 10   
## Y dimension: 25 1  
## Fit method: svdpc  
## Number of components considered: 3  
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps  
## X 61.88 84.4 91.17  
## .outcome 80.82 81.2 85.74

predictions <- pcr\_model %>% predict(test.data)  
  
data.frame(  
 RMSE = caret::RMSE(predictions, test.data$mpg),  
 Rsquare = caret::R2(predictions, test.data$mpg)  
)

## RMSE Rsquare  
## 1 2.197363 0.8628547

#### 3.2 Compare the RMSE of the two methods based on the test data set. Which one is better?

set.seed(123)  
  
  
pls\_model <- train(  
 mpg~., data = train.data, method = "pls",  
 scale = TRUE,  
 trControl = trainControl("cv", number = 10),  
 tuneLength = 10  
 )  
  
plot(pls\_model)



pls\_model$bestTune

## ncomp  
## 1 1

summary(pls\_model$finalModel)

## Data: X dimension: 25 10   
## Y dimension: 25 1  
## Fit method: oscorespls  
## Number of components considered: 1  
## TRAINING: % variance explained  
## 1 comps  
## X 61.81  
## .outcome 82.12

predictions <- pls\_model %>% predict(test.data)  
  
data.frame(  
 RMSE = caret::RMSE(predictions, test.data$mpg),  
 Rsquare = caret::R2(predictions, test.data$mpg)  
)

## RMSE Rsquare  
## 1 1.441384 0.9125933

#RESULTS  
#Comparting the two models shows that PLS is better for this data set since it has a lower RMSE and higher RSQ.

#### Q4: So far, you have performed multiple model fitting in Q1~Q3, and used different model assessment/selection methods to evaluate different models.

#### Based on what we covered in the lecture of Week 5, please summarize the model assessment tools (i.e. which type of error) you used in solving each question of Q1~Q3.

#### Question 1 Models ####  
  
  
### Models  
  
## Linear Regression  
# Linear Regression is a predictive analysis tool using dependent and independent variables but strictly graphing and predicting y values through linear patterns only. This means y = mx + b is the basis of the model.  
  
# RSQ, for my model I used R squared which is the measure of variation when using the dependent variable to predict the independent variable. R squared is a percentage, so the closer the R squared is to 1 then the lower the variation meaning the better model.   
  
  
# Multiple Linear Regression  
# Multiple Linear Regression is the same as linear regression but uses explanatory variables to allow for a better prediction of the model. This splits the data, like in question two when we split the data based on species.   
# RSQ and adjRSQ, We already discussed R squared, so I am going to explain what adjusted R squared is. Adjusted R squared is the same value as R squared but punishes data sets that are too complicated since the base R squared slowly gets better as you add variables to the model.  
  
# Forward and Backward Step-wise Selection  
# Step-wise regression is a model that changes the amount of variables in the model to optimize the amount of variables for accuracy and simplicity. A forward step-wise model starts off with an empty model and adds variables and backwards is the opposite.  
# RSQ and RSE, For this model I used RSE and RSQ to determine the best model (also the model's AIC score). RSE is residual standard error and as the name implies, we would like to decrease error in the model so the closer the number is to zero the better.

#### Question 2 Models ####  
  
  
### Models  
  
## Ridge Regression  
# Ridge regression is a type of variable selection for a model allowing the "shrinking" of variables. This changes the weight of variables in the model allowing for more accurate results.  
# RSQ and SSR, SSR is also known as the residual sum of squares. This shows the error in the model, with the smaller the number the tighter the fit the model has on the data (not between 0 and 1). This is difficult to use by itself since the SSR can be extremely low but the model be too tight to accurately predict future data.  
  
# LASSO  
# The LASSO is very similar to Ridge Regression and accomplishes the same thing; however, LASSO can set variables equal to zero rather than just shrinking the variables.  
# RSQ and SSR, explained above.  
  
# Elastic Net  
# Elastic Net is an in between Ridge Regression and LASSO and can be changed to be more like one model rather than the other variable selection model.  
# RSQ and SSR  
  
  
### Validation Criteria  
## AIC/BIC  
# AIC and BIC both accomplish the same task, checking the "goodness" of the fit while also punishing models that are too complicated. BIC does a better (more harsh) job for punishing over complex models.  
## Cross-Validation  
# Cross-Validation is using the training data to both train and test the data. You split the data into k subgroups and can make a model from each subgroup k and test the models on each other subgroup. This is very good way to make a model with the amount of data is limited.

#### Question 3 Models ####  
  
  
### Models  
  
## PCR  
# Principle Component Analysis is using hyper planes to find principle components to shrink the amount and weight of variables. PCR typically ends up reducing the dimensions of the model.This model relies on variance to check the components.  
# RMSE and RSQ, Residual Mean Square Error is a very common measure of differences in models. The lower the RMSE the better the fit the model has. This can also be referred to as the "cost" function when minimizing.  
  
# PLS  
# Rather than looking at variance, Partial Least Square Regression rather uses the observable variables (x) and predicted variables (y) to form several individual linear regression models. This model then tries to find the multidimensional direction of x (2D) to with the highest variance in y.   
# RMSE and RSQ, explained above.