# FE590. Assignment #3.

## John-Craig Borman (10402229)

## 2018-11-12

# Instructions

In this assignment, you should use R markdown to answer the questions below. Simply type your R code into embedded chunks as shown above.

When you have completed the assignment, knit the document into a PDF file, and upload *both* the .pdf and .Rmd files to Canvas.

Note that you must have LaTeX installed in order to knit the equations below. If you do not have it installed, simply delete the questions below.

# Question 1 (based on JWHT Chapter 5, Problem 8)

In this problem, you will perform cross-validation on a simulated data set.

You will use this personalized simulated data set for this problem:

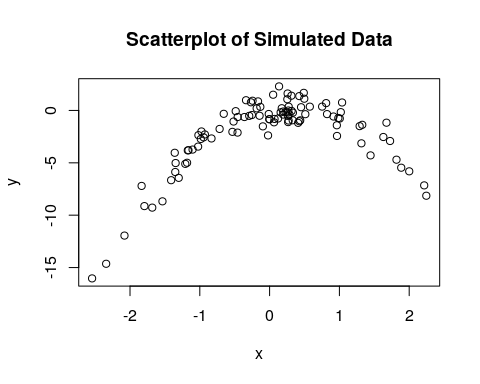
CWID = 10402229 #Place here your Campus wide ID number, this will personalize  
#your results, but still maintain the reproduceable nature of using seeds.  
#If you ever need to reset the seed in this assignment, use this as your seed  
#Papers that use -1 as this CWID variable will earn 0's so make sure you change  
#this value before you submit your work.  
personal = CWID %% 10000  
set.seed(personal)  
  
  
y <- rnorm(100)  
x <- rnorm(100)  
y <- x - 2\*x^2 + rnorm(100)

1. In this data set, what is *n* and what is *p*?

*n* - the number of observations in the data set (100) *p* - The number of parameters or variables in the data set (1)

1. Create a scatterplot of *x* against *y*. Comment on what you find.

plot(x = x, y = y, main = "Scatterplot of Simulated Data")



We notice that the data approximately follows a concave (downward) parabola such that , where is proportional with from -2.5 to about 0, and then becomes inversely proportional from 0 to 2.5.

1. Compute the LOOCV errors that result from fitting the following four models using least squares:

First, let’s do some setup. Define a data.frame to be subset per each model build in LOOCV and a vector m.cv to store each model’s CV error.

library(boot)  
df = data.frame(x=x, y=y)  
m.cv = c()

Calculate the CV Errors for each model

# Model 1  
m1 = glm(y ~ poly(x, 1), data = df)  
m.cv[1] = cv.glm(data = df, glmfit = m1)$delta[1]  
  
# Model 2  
m2 = glm(y ~ poly(x, 2), data = df)  
m.cv[2] = cv.glm(data = df, glmfit = m2)$delta[1]  
  
# Model 3  
m3 = glm(y ~ poly(x, 3), data = df)  
m.cv[3] = cv.glm(data = df, glmfit = m3)$delta[1]  
  
# Model 4  
m4 = glm(y ~ poly(x, 4), data = df)  
m.cv[4] = cv.glm(data = df, glmfit = m4)$delta[1]

1. Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

m.cv\_min = which.min(m.cv)  
m.cv

## [1] 9.6816227 0.9922244 1.0075138 1.0432976

m.cv\_min

## [1] 2

m.cv[m.cv\_min]

## [1] 0.9922244

The second model minimizes the LOOCV error. This is not surprising since we know that the relationship between and is downward parabolic, hence can be modeled by a quadratic fit. Model two fits a quadratic function to the data via OLS, thus it make sense that this best models the data

1. Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawnbased on the cross-validation results?

summary(m1)

##   
## Call:  
## glm(formula = y ~ poly(x, 1), data = df)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -10.2467 -1.0363 0.8618 1.7251 4.1663   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.0748 0.3008 -6.898 5.21e-10 \*\*\*  
## poly(x, 1) 14.7527 3.0077 4.905 3.71e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 9.046258)  
##   
## Null deviance: 1104.18 on 99 degrees of freedom  
## Residual deviance: 886.53 on 98 degrees of freedom  
## AIC: 508  
##   
## Number of Fisher Scoring iterations: 2

Model 1: is statistically significant at the highest degree (~0), but also has the highest LOOCV error.

summary(m2)

##   
## Call:  
## glm(formula = y ~ poly(x, 2), data = df)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.37172 -0.65502 -0.00739 0.57782 2.44563   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.07482 0.09829 -21.11 <2e-16 \*\*\*  
## poly(x, 2)1 14.75271 0.98295 15.01 <2e-16 \*\*\*  
## poly(x, 2)2 -28.15694 0.98295 -28.64 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.9661872)  
##   
## Null deviance: 1104.18 on 99 degrees of freedom  
## Residual deviance: 93.72 on 97 degrees of freedom  
## AIC: 285.3  
##   
## Number of Fisher Scoring iterations: 2

Model 2: and are statistically significant at the highest degree with the lowest LOOCV error.

summary(m3)

##   
## Call:  
## glm(formula = y ~ poly(x, 3), data = df)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.37204 -0.69417 -0.00868 0.58942 2.44098   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.07482 0.09874 -21.01 <2e-16 \*\*\*  
## poly(x, 3)1 14.75271 0.98739 14.94 <2e-16 \*\*\*  
## poly(x, 3)2 -28.15694 0.98739 -28.52 <2e-16 \*\*\*  
## poly(x, 3)3 0.35592 0.98739 0.36 0.719   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.9749321)  
##   
## Null deviance: 1104.176 on 99 degrees of freedom  
## Residual deviance: 93.593 on 96 degrees of freedom  
## AIC: 287.17  
##   
## Number of Fisher Scoring iterations: 2

Model 3: and are statistically significant at the highest degree, while is not statistically significant with the 2nd lowest LOOCV error

summary(m4)

##   
## Call:  
## glm(formula = y ~ poly(x, 4), data = df)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.35467 -0.69598 0.00338 0.60461 2.41284   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.07482 0.09923 -20.910 <2e-16 \*\*\*  
## poly(x, 4)1 14.75271 0.99229 14.867 <2e-16 \*\*\*  
## poly(x, 4)2 -28.15694 0.99229 -28.376 <2e-16 \*\*\*  
## poly(x, 4)3 0.35592 0.99229 0.359 0.721   
## poly(x, 4)4 -0.23114 0.99229 -0.233 0.816   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.9846322)  
##   
## Null deviance: 1104.18 on 99 degrees of freedom  
## Residual deviance: 93.54 on 95 degrees of freedom  
## AIC: 289.11  
##   
## Number of Fisher Scoring iterations: 2

Model 4: and are statistically significant at the highest degree, while and are not statistically significant with the 2nd highest LOOCV error

*Conclusion:* The statistical significance of the coefficient estimates are consistent and agree with the LOOCV results. We find that only and are ever statistically significant (again indiciative of a quadratic relationship between and ). Further, LOOCV shows that the model with the lowest error is Model 2 where all predictors (, ) are statistically significant at the highest degree.

# Question 2 (based on JWTH Chapter 7, Problem 10)

The question refers to the ‘College’ data set

Load ISLR and checkout the data set:

library(ISLR)  
attach(College)  
dim(College)

## [1] 777 18

names(College)

## [1] "Private" "Apps" "Accept" "Enroll" "Top10perc"   
## [6] "Top25perc" "F.Undergrad" "P.Undergrad" "Outstate" "Room.Board"   
## [11] "Books" "Personal" "PhD" "Terminal" "S.F.Ratio"   
## [16] "perc.alumni" "Expend" "Grad.Rate"

1. Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform subset selection (your choice on how) in order to identify a satisfactory model that uses just a subset of the predictors (if your approach suggests using all of the predictors, then follow your results and use them all).

Split data into non-overlapping test and train subsets:

obs <- 1:nrow(College)  
train.idx <- sample(x = obs, size = round(nrow(College) / 2))  
test.idx <- obs[!(obs %in% train.idx)]  
  
intersect(train.idx, test.idx) # Confirm that training and testing sets do not overlap

## integer(0)

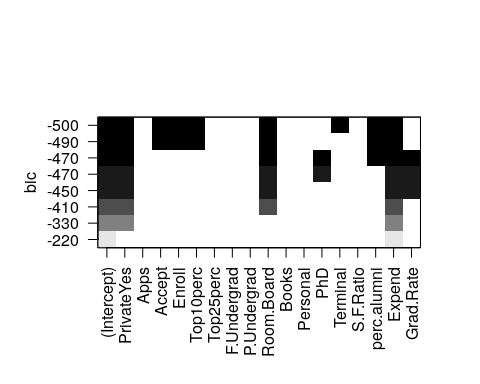
# Subset College  
train <- College[train.idx,]  
test <- College[test.idx,]

Perform an exhaustive feature subset selection with the Leaps package:

library(leaps)  
out <- regsubsets(Outstate~., data=train)  
summary(out)

## Subset selection object  
## Call: regsubsets.formula(Outstate ~ ., data = train)  
## 17 Variables (and intercept)  
## Forced in Forced out  
## PrivateYes FALSE FALSE  
## Apps FALSE FALSE  
## Accept FALSE FALSE  
## Enroll FALSE FALSE  
## Top10perc FALSE FALSE  
## Top25perc FALSE FALSE  
## F.Undergrad FALSE FALSE  
## P.Undergrad FALSE FALSE  
## Room.Board FALSE FALSE  
## Books FALSE FALSE  
## Personal FALSE FALSE  
## PhD FALSE FALSE  
## Terminal FALSE FALSE  
## S.F.Ratio FALSE FALSE  
## perc.alumni FALSE FALSE  
## Expend FALSE FALSE  
## Grad.Rate FALSE FALSE  
## 1 subsets of each size up to 8  
## Selection Algorithm: exhaustive  
## PrivateYes Apps Accept Enroll Top10perc Top25perc F.Undergrad  
## 1 ( 1 ) " " " " " " " " " " " " " "   
## 2 ( 1 ) "\*" " " " " " " " " " " " "   
## 3 ( 1 ) "\*" " " " " " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " " " " " " "   
## 5 ( 1 ) "\*" " " " " " " " " " " " "   
## 6 ( 1 ) "\*" " " " " " " " " " " " "   
## 7 ( 1 ) "\*" " " "\*" "\*" "\*" " " " "   
## 8 ( 1 ) "\*" " " "\*" "\*" "\*" " " " "   
## P.Undergrad Room.Board Books Personal PhD Terminal S.F.Ratio  
## 1 ( 1 ) " " " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " " " "   
## 3 ( 1 ) " " "\*" " " " " " " " " " "   
## 4 ( 1 ) " " "\*" " " " " " " " " " "   
## 5 ( 1 ) " " "\*" " " " " "\*" " " " "   
## 6 ( 1 ) " " "\*" " " " " "\*" " " " "   
## 7 ( 1 ) " " "\*" " " " " " " " " " "   
## 8 ( 1 ) " " "\*" " " " " " " "\*" " "   
## perc.alumni Expend Grad.Rate  
## 1 ( 1 ) " " "\*" " "   
## 2 ( 1 ) " " "\*" " "   
## 3 ( 1 ) " " "\*" " "   
## 4 ( 1 ) " " "\*" "\*"   
## 5 ( 1 ) " " "\*" "\*"   
## 6 ( 1 ) "\*" "\*" "\*"   
## 7 ( 1 ) "\*" "\*" " "   
## 8 ( 1 ) "\*" "\*" " "

plot(out)



Given the results of the Linear Regression model subsets, we find that the model that minimizes BIC on the training data set contains the following features: - PrivateYes (Private) - Accept - Enroll - Top10perc  
- Room.Board  
- Terminal - perc.alumni - Expend

1. Fit a GAM on the training data, using out-of-state tuition as the response and the features selected in the previous step as the predictors, using splines of each feature with 5 df.

library(gam)

## Loading required package: splines

## Loading required package: foreach

## Loaded gam 1.15

lm.m1 <- lm(Outstate ~ Private + Accept + Enroll + Top10perc + Room.Board + Terminal + perc.alumni + Expend, data = train)  
summary(lm.m1)

##   
## Call:  
## lm(formula = Outstate ~ Private + Accept + Enroll + Top10perc +   
## Room.Board + Terminal + perc.alumni + Expend, data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -7109.0 -1228.6 25.9 1146.8 9603.7   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.858e+03 6.839e+02 -2.717 0.006888 \*\*   
## PrivateYes 2.540e+03 3.331e+02 7.625 1.98e-13 \*\*\*  
## Accept 7.950e-01 1.201e-01 6.618 1.25e-10 \*\*\*  
## Enroll -2.019e+00 3.078e-01 -6.559 1.78e-10 \*\*\*  
## Top10perc 3.697e+01 8.505e+00 4.347 1.78e-05 \*\*\*  
## Room.Board 8.379e-01 1.181e-01 7.093 6.46e-12 \*\*\*  
## Terminal 3.326e+01 8.744e+00 3.803 0.000166 \*\*\*  
## perc.alumni 5.031e+01 1.014e+01 4.961 1.06e-06 \*\*\*  
## Expend 2.079e-01 2.772e-02 7.503 4.48e-13 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1941 on 379 degrees of freedom  
## Multiple R-squared: 0.7595, Adjusted R-squared: 0.7544   
## F-statistic: 149.6 on 8 and 379 DF, p-value: < 2.2e-16

gam.m2 <- gam(Outstate ~ Private + s(Accept, 5) + s(Enroll, 5) + s(Top10perc, 5) + s(Room.Board, 5) + s(Terminal, 5) + s(perc.alumni, 5) + s(Expend, 5), data = train)  
summary(gam.m2)

##   
## Call: gam(formula = Outstate ~ Private + s(Accept, 5) + s(Enroll, 5) +   
## s(Top10perc, 5) + s(Room.Board, 5) + s(Terminal, 5) + s(perc.alumni,   
## 5) + s(Expend, 5), data = train)  
## Deviance Residuals:  
## Min 1Q Median 3Q Max   
## -6642.01 -929.44 76.04 1063.44 7633.87   
##   
## (Dispersion Parameter for gaussian family taken to be 3060585)  
##   
## Null Deviance: 5933703550 on 387 degrees of freedom  
## Residual Deviance: 1074265203 on 350.9999 degrees of freedom  
## AIC: 6932.649   
##   
## Number of Local Scoring Iterations: 3   
##   
## Anova for Parametric Effects  
## Df Sum Sq Mean Sq F value Pr(>F)   
## Private 1 1606802426 1606802426 524.998 < 2.2e-16 \*\*\*  
## s(Accept, 5) 1 367164734 367164734 119.966 < 2.2e-16 \*\*\*  
## s(Enroll, 5) 1 266670811 266670811 87.131 < 2.2e-16 \*\*\*  
## s(Top10perc, 5) 1 1022127223 1022127223 333.965 < 2.2e-16 \*\*\*  
## s(Room.Board, 5) 1 305284128 305284128 99.747 < 2.2e-16 \*\*\*  
## s(Terminal, 5) 1 59402742 59402742 19.409 1.403e-05 \*\*\*  
## s(perc.alumni, 5) 1 90608719 90608719 29.605 9.957e-08 \*\*\*  
## s(Expend, 5) 1 381354009 381354009 124.602 < 2.2e-16 \*\*\*  
## Residuals 351 1074265203 3060585   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Anova for Nonparametric Effects  
## Npar Df Npar F Pr(F)   
## (Intercept)   
## Private   
## s(Accept, 5) 4 8.9574 6.738e-07 \*\*\*  
## s(Enroll, 5) 4 4.7952 0.0008891 \*\*\*  
## s(Top10perc, 5) 4 2.0622 0.0853216 .   
## s(Room.Board, 5) 4 1.9923 0.0952130 .   
## s(Terminal, 5) 4 2.5272 0.0405241 \*   
## s(perc.alumni, 5) 4 0.5370 0.7086589   
## s(Expend, 5) 4 15.4257 1.236e-11 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

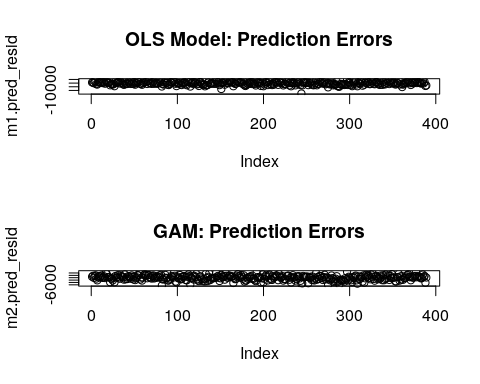
Private is a categorical variable (unordered), splines cannot be applied to such data. Splines, however, are applied to all other selected features with Degrees of Freedom (DF =) 5

Both an OLS model and a GAM are fit above so as to benchmark the GAM to some other model solely for reference

1. Evaluate the model obtained on the test set, and explain the results obtained

Evaluating lm.m1 and gam.m2 on test:

# Predict on test data  
m1.pred <- predict(lm.m1, newdata = test)  
m2.pred <- predict(gam.m2, newdata = test)  
  
# Get the prediction errors (residuals)  
m1.pred\_resid <- test$Outstate - m1.pred  
m2.pred\_resid <- test$Outstate - m2.pred  
  
par(mfrow=c(2,1))  
# Plot the residuals  
plot(m1.pred\_resid, main="OLS Model: Prediction Errors")  
plot(m2.pred\_resid, main="GAM: Prediction Errors")



print(paste("OLS Residual Mean: ", mean(m1.pred\_resid)))

## [1] "OLS Residual Mean: -104.034603931169"

print(paste("GAM Residual Mean: ", mean(m2.pred\_resid)))

## [1] "GAM Residual Mean: -3.28994397011293"

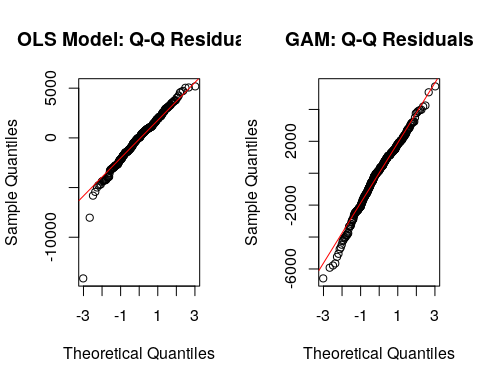
print(paste("OLS Residual Variance: ", var(m1.pred\_resid)))

## [1] "OLS Residual Variance: 4809283.08361994"

print(paste("GAM Residual Variance: ", var(m2.pred\_resid)))

## [1] "GAM Residual Variance: 3820180.07761222"

par(mfrow=c(1,2))  
  
{  
 qqnorm(m1.pred\_resid, main = "OLS Model: Q-Q Residuals")  
 qqline(m1.pred\_resid, col = "red")  
}  
  
{  
 qqnorm(m2.pred\_resid, main = "GAM: Q-Q Residuals")  
 qqline(m2.pred\_resid, col = "red")  
}



# check the MSE  
print(paste("OLS MSE: ", sum(m1.pred\_resid^2)/length(m1.pred\_resid)))

## [1] "OLS MSE: 4807743.08684734"

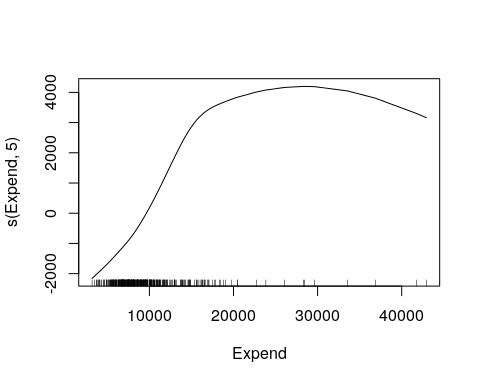
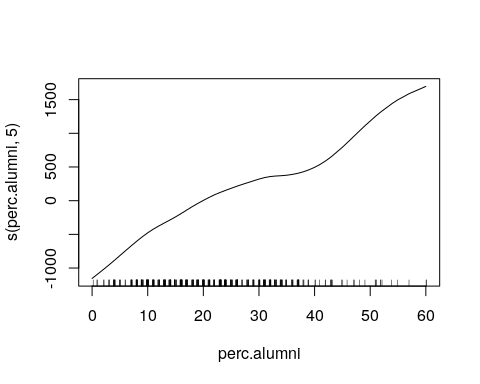
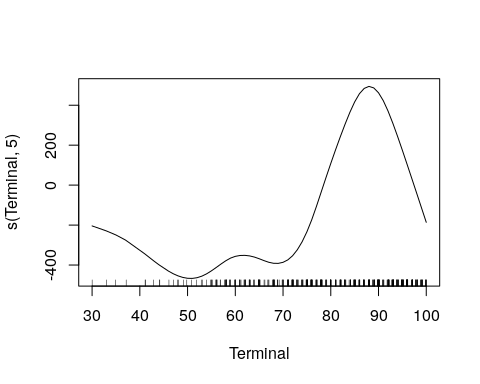
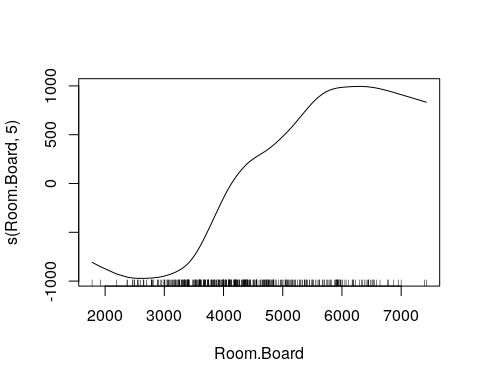
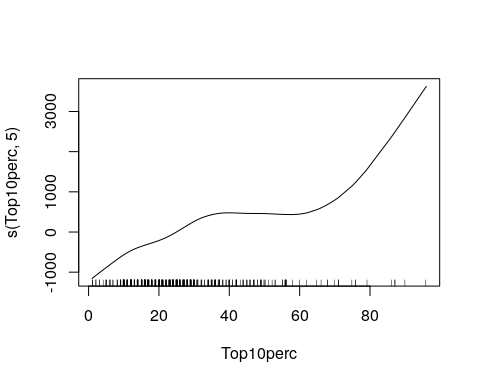
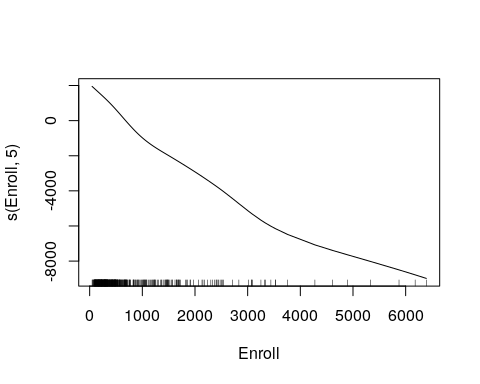
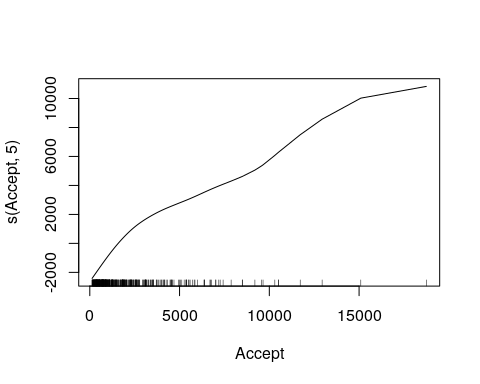
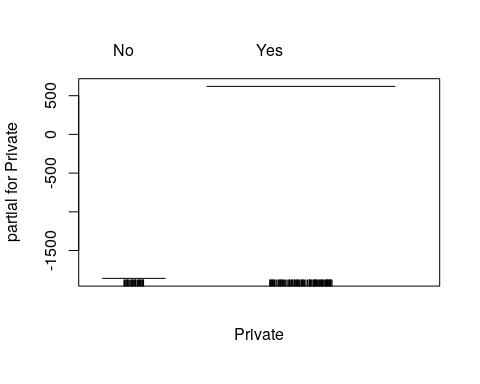
print(paste("GAM MSE: ", sum(m2.pred\_resid^2)/length(m2.pred\_resid)))

## [1] "GAM MSE: 3810370.38700521"

We find that the model that minimizes the MSE on the testing data is the gam.m2 model. Comparatively, the results show that the residuals of the GAM model most closely resemble a white noise error term with mean 0, smaller variance term. Further, the GAM’s MSE is less than that of the OLS model. Therefore, the GAM shows better out of sample performance than the benchmark OLS

1. For which variables, if any, is there evidence of a non-linear relationship with the response? Which are probably linear? Justify your answers.

par(mfrow=c(1,1))  
plot(gam.m2)



plot.gam() plots the partial terms (primary effects) of each compnent on the outcome variable. Terms like Accept, Enroll, and perc.alumni show approximately linear relationship with the response as their plots above show roughly constant proportional relationships throughout the domain of the input feature. Expend most closely resembles a concave parabola. Top10perc is roughly similar to a nonlinear polynomial of the 3rd degree as it increases proportionally from [0, 40], flattens from [40, 60] and then increases again from [60, 100]. Room.Board also appears to be a nonlinear polynomial of the 3rd degree. For [2000, 4500], the relationship is a convex parabola while from [4500, 7000] is a concave parabola. Lastly, Terminal is a higher order polynomial (>= 5) since the relationship varies throughout the domain of the feature.

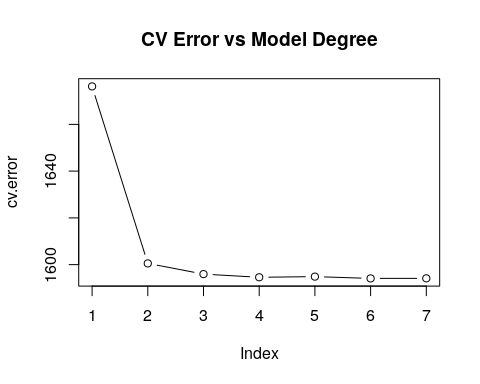
# Question 3 (based on JWHT Chapter 7, Problem 6)

In this exercise, you will further analyze the Wage data set.

1. Perform polynomial regression to predict wage using age. Use cross-validation to select the optimal degree d for the polynomial. What degree was chosen? Make a plot of the resulting polynomial fit to the data.

Testing polynomial models of degrees 1 through 7:

cv.error <- rep(Inf, 7)  
  
for(i in 1:7){  
 mdl <- glm(wage ~ poly(age, i), data = Wage)  
 cv.error[i] = cv.glm(data = Wage, glmfit = mdl)$delta[1]  
}  
  
plot(cv.error, type="b", main = "CV Error vs Model Degree")



cv.error

## [1] 1676.235 1600.529 1595.960 1594.596 1594.879 1594.119 1594.145

idx <- which.min(cv.error)  
idx

## [1] 6

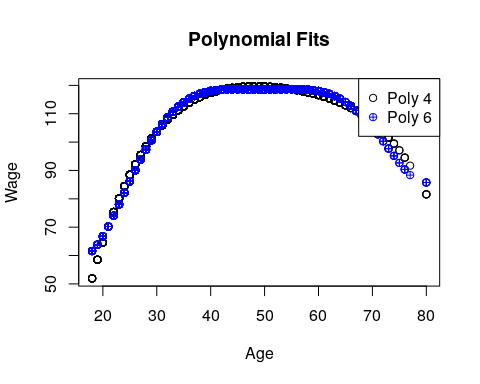
cv.error[idx]

## [1] 1594.119

The model that best minimizes the error via CV is the polynomial model of degree = 6. However, from the plot above we can see that this model is likely unnecessarily complicated as each extra term beyond the 3rd degree is only marginally decreasing the CV error.

Let’s compare the plots of models of degree 4 and 6

m4 <- glm(wage ~ poly(age, 4), data = Wage)  
m6 <- glm(wage ~ poly(age, 6), data = Wage)  
  
{  
 plot(y=m4$fitted.values, x=Wage$age, pch = 1, ylab = "Wage", xlab = "Age", main = "Polynomial Fits")  
 points(y=m6$fitted.values, x=Wage$age, col = "blue", pch = 10)  
 legend("topright", c("Poly 4", "Poly 6"), col = c("black", "blue"), pch = c(1, 10))  
}

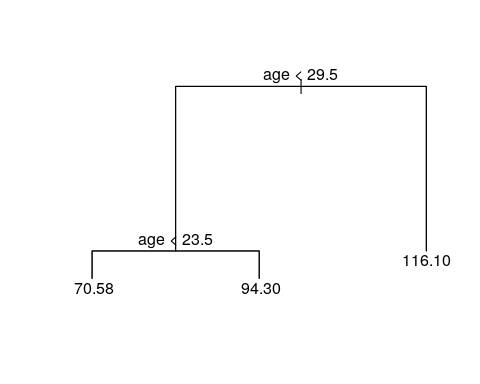


1. Fit a step function to predict wage using age, and perform cross-validation to choose the optimal number of cuts. Make a plot of the fit obtained.

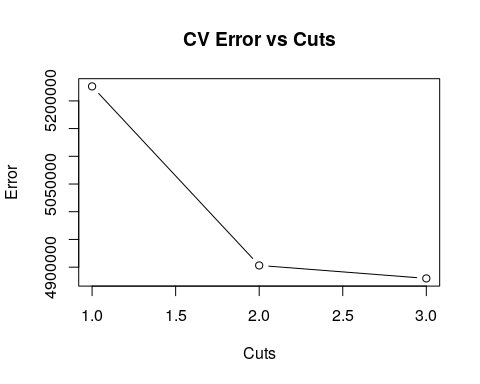
library(tree)  
  
# Model regression tree  
mdl <- tree(wage~age, data = Wage)  
summary(mdl)

##   
## Regression tree:  
## tree(formula = wage ~ age, data = Wage)  
## Number of terminal nodes: 3   
## Residual mean deviance: 1615 = 4839000 / 2997   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -96.000 -25.610 -6.254 0.000 15.530 202.300

# Plot the Tree  
{  
 plot(mdl)  
 text(mdl, pretty = 1)  
}



# Run K-Fold CV  
cv.mdl <- cv.tree(mdl)  
plot(cv.mdl$size, cv.mdl$dev, type="b", main="CV Error vs Cuts", ylab="Error", xlab="Cuts")



According to the plot above. Three cuts minimizes CV error, however this improvement is marginal beyond the second cut.

# Question 4 (based on JWHT Chapter 8, Problem 8)

In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable.

1. Split the data set into a training set and a test set.

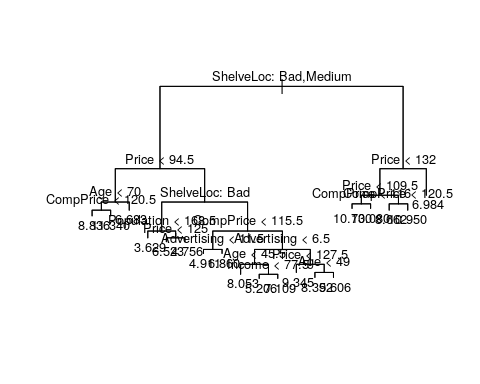
obs <- 1:nrow(Carseats)  
train.idx <- sample(obs, size = round(length(obs)/2))  
test.idx <- obs[!(obs %in% train.idx)]  
  
intersect(train.idx, test.idx) # Should be Null

## integer(0)

train <- Carseats[train.idx,]  
test <- Carseats[test.idx,]

1. Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

library(rpart)  
mdl <- tree(Sales~., data = train)  
  
{  
 plot(mdl)  
 text(mdl, pretty = 0, cex = 0.8)  
}



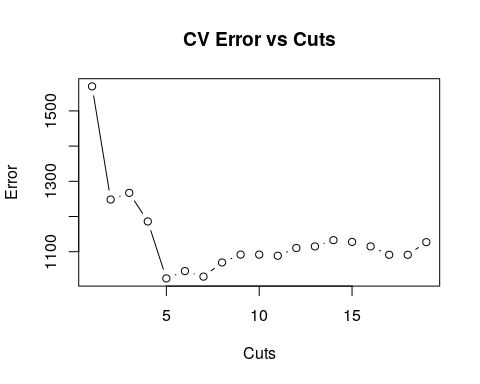
summary(mdl)

##   
## Regression tree:  
## tree(formula = Sales ~ ., data = train)  
## Variables actually used in tree construction:  
## [1] "ShelveLoc" "Price" "Age" "CompPrice" "Population"   
## [6] "Advertising" "Income"   
## Number of terminal nodes: 19   
## Residual mean deviance: 1.943 = 351.6 / 181   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -4.75100 -0.78590 0.03508 0.00000 0.82610 3.62600

The MSE (residual mean deviance in the summary above) is 2.053. The ShelveLoc and Price variables have the most significant impact on the outcome as they are the first and second variables to partition the space, respectively.

1. Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

cv.mdl <- cv.tree(mdl)  
plot(cv.mdl$size, cv.mdl$dev, type="b", main="CV Error vs Cuts", ylab="Error", xlab="Cuts")



Above we can see that the number of cuts locally minimizes the CV error at nCuts = 6. Error then increases, plateaus and steadily declines as the number of cuts increase. Thus, let’s examine the tree with 6 cuts:

mdl.pruned <- prune.tree(mdl, best = 6)  
mean( (test$Sales - predict(mdl, newdata = test))^2) # MSE Non-Pruned Tree

## [1] 4.905835

mean( (test$Sales - predict(mdl.pruned, newdata = test))^2) # MSE Pruned Tree

## [1] 5.251832

Pruning does not improve the MSE of the model, which is intuitive since we are removing decision-making from the model. Though MSE is increased, we are reducing the amount of overfitting in the model.

1. Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

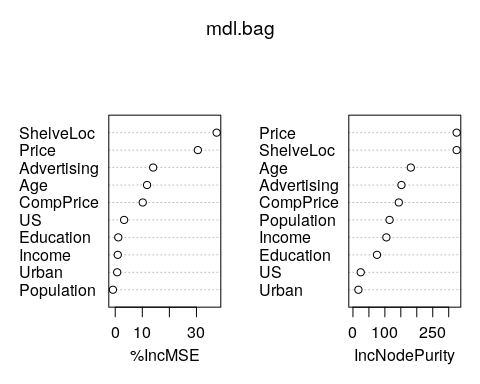
mdl.bag <- randomForest(Sales ~ ., data = train, importance=TRUE)  
mdl.bag

##   
## Call:  
## randomForest(formula = Sales ~ ., data = train, importance = TRUE)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## Mean of squared residuals: 3.357296  
## % Var explained: 56.6

importance(mdl.bag)

## %IncMSE IncNodePurity  
## CompPrice 10.1626771 143.65510  
## Income 0.9343679 104.73854  
## Advertising 14.0104534 151.93162  
## Population -0.8421888 114.69947  
## Price 30.5231982 324.91038  
## ShelveLoc 37.4624584 324.88664  
## Age 11.7611060 181.25809  
## Education 1.1258044 75.36676  
## Urban 0.7189220 17.07740  
## US 3.2821644 24.38017

varImpPlot(mdl.bag)



The variable importance plots affirm that Price and ShelveLoc are again the most important features as removing them would subject the model to a high % increase in MSE.

The new MSE is:

mean((test$Sales - predict(mdl.bag, newdata = test))^2)

## [1] 3.246154

# Question 5 (based on JWTH Chapter 8, Problem 10)

Use boosting (and bagging) to predict Salary in the Hitters data set

1. Remove the observations for which salary is unknown, and then log-transform the salaries

library(gbm)

## Loading required package: survival

##   
## Attaching package: 'survival'

## The following object is masked from 'package:rpart':  
##   
## solder

## The following object is masked from 'package:boot':  
##   
## aml

## Loading required package: lattice

##   
## Attaching package: 'lattice'

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

## Loading required package: parallel

## Loaded gbm 2.1.3

df <- Hitters[!is.na(Hitters$Salary),]  
df$Salary <- log(df$Salary)

1. Split the data into training and testing sets for cross validation purposes.

obs <- 1:nrow(df)  
train.idx <- sample(obs, size = round(length(obs)/2))  
test.idx <- obs[!(obs %in% train.idx)]  
  
intersect(train.idx, test.idx) # Should be Null

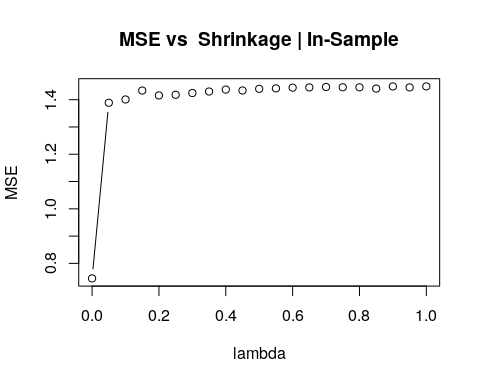
## integer(0)

train <- df[train.idx,]  
test <- df[test.idx,]

1. Perform boosting on the training set with 1000 trees for a range of values of the shrinkage parameter . Produce a plot with different shrinkage parameters on the x-axis and the corresponding training set MSE on the y-axis

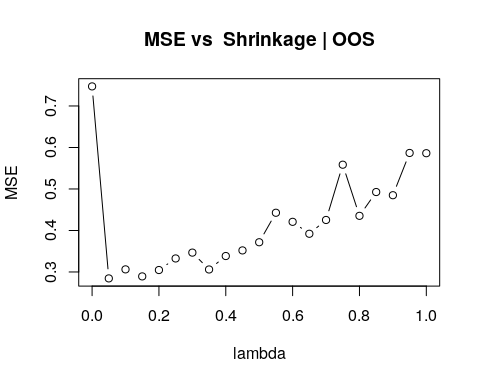
lambda <- seq(0, 1, by = 0.05)  
MSE <- rep(Inf, 20)  
  
for(i in 1:length(lambda)){  
 mdl.boost <- gbm(Salary~., data = train, n.trees = 1000, shrinkage = lambda[i], distribution = "gaussian")  
 MSE[i] <- mean( (test$Salary - predict(mdl.boost, newdata = train, n.trees = 1000))^2)  
}

plot(y = MSE, x = lambda, main = "MSE vs Shrinkage | In-Sample", type = "b")



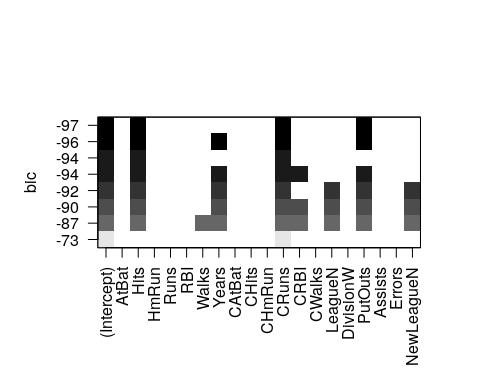
1. Produce a plot similar to the last one, but this time using the test set MSE

MSE <- rep(Inf, 20)  
  
for(i in 1:length(lambda)){  
 mdl.boost <- gbm(Salary~., data = train, n.trees = 1000, shrinkage = lambda[i], distribution = "gaussian")  
 MSE[i] <- mean( (test$Salary - predict(mdl.boost, newdata = test, n.trees = 1000))^2)  
}  
  
plot(y = MSE, x = lambda, main = "MSE vs Shrinkage | OOS", type = "b")



1. Fit the model using two other regression techniques (from previous classes) and compare the MSE of those techniques to the results of these boosted trees.

rsubs <- regsubsets(Salary ~ ., data = train)  
plot(rsubs)



Utilizing regression subsets to select features based on the best linear regression model, the best model is:

Let’s build an OLS and GAM from this selected model:

mdl.lm <- lm(Salary ~ Runs + CHits, data = train)  
mdl.gam <- gam(Salary ~ s(Runs, 5) + s(CHits, 5), data = train)

Now testing on OOS, we have MSEs

mean((test$Salary - predict(mdl.lm, newdata = test))^2) # OLS MSE

## [1] 0.4652213

mean((test$Salary - predict(mdl.gam, newdata = test))^2)# GAM MSE

## [1] 0.2897559

min(MSE) # GBM MSE

## [1] 0.2845555

Despite utilizing feature selection before building OLS and GAMs, the OOS MSE of the boosted trees is still superior. This is given that we use the MSE of the error minimizing -shrinkage parameter.

1. Reproduce (c) and (d), but this time use bagging instead of boosting and compare to the boosted MSE’s and the MSE’s from (e)

mdl.bag <- randomForest(Salary ~ ., data = train, importance=TRUE)  
mdl.bag

##   
## Call:  
## randomForest(formula = Salary ~ ., data = train, importance = TRUE)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 6  
##   
## Mean of squared residuals: 0.1970462  
## % Var explained: 76.24

Now we have:

mean((test$Salary - predict(mdl.lm, newdata = test))^2) # OLS MSE

## [1] 0.4652213

mean((test$Salary - predict(mdl.gam, newdata = test))^2)# GAM MSE

## [1] 0.2897559

min(MSE) # GBM MSE

## [1] 0.2845555

mean((test$Salary - predict(mdl.bag, newdata = test))^2)# RF MSE

## [1] 0.2203611

Finally, we see that the bagging method (randomForest) outdoes the previously superior boosting method with the lowest MSE.