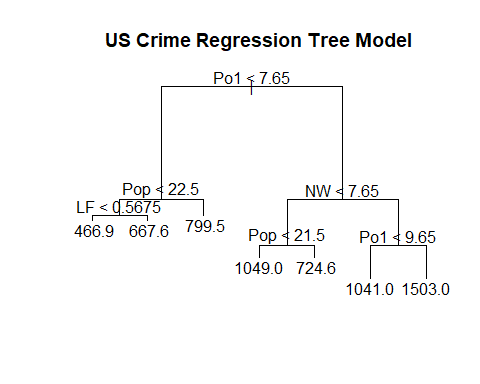
hw7

2023-02-26

## Question 10.1

**Using the same crime data set uscrime.txt as in Questions 8.2 and 9.1, find the best model you can using**  
**(a) a regression tree model, and (b) a random forest model.**  
**In R, you can use the tree package or the rpart package, and the randomForest package. For each model, describe one or two qualitative takeaways you get from analyzing the results (i.e., don’t just stop when you have a good model, but interpret it too).**

data2 <- read.delim2('uscrime.txt')  
data2 <- as.data.frame(lapply(data2, as.numeric))  
  
set.seed(2)  
tree\_model <- tree(Crime ~ ., data= data2)  
  
plot(tree\_model)  
text(tree\_model)  
title('US Crime Regression Tree Model')



summary(tree\_model)

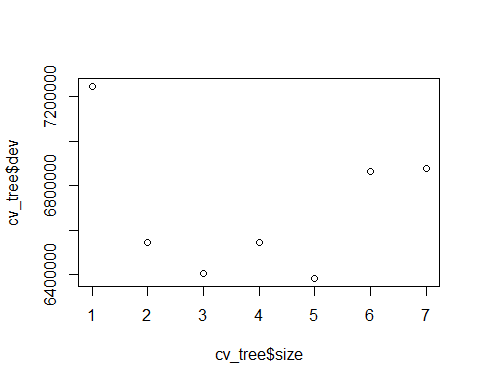
##   
## Regression tree:  
## tree(formula = Crime ~ ., data = data2)  
## Variables actually used in tree construction:  
## [1] "Po1" "Pop" "LF" "NW"   
## Number of terminal nodes: 7   
## Residual mean deviance: 47390 = 1896000 / 40   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -573.900 -98.300 -1.545 0.000 110.600 490.100

Using the ‘Tree’ library, we are able to generate a regression tree model. Without any pruning or CV, we are able to generate a tree with 7 terminal nodes or leaves, and also a unique set of criteria per split. Out of the 16 variables available, the model uses Po1(per capita expenditure on police protection in 1960), Pop (state population in 1960 in hundred thousands), LF(labour force participation rate of civilian urban males in the age-group 14-24), and NW(percentage of nonwhites in the population) as its 4 variables. Both Po1 and Pop are used twice in splits. The ‘Residual mean deviance’ is similar to the mean squared error for other models, and our ‘Residual mean deviance’ is 47390. We want to minimize error, so lets see if tuning our model can return better results.

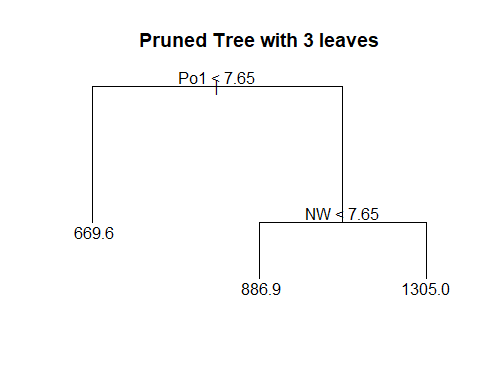
cv\_tree = cv.tree(tree\_model)  
summary(cv\_tree)

## Length Class Mode   
## size 7 -none- numeric   
## dev 7 -none- numeric   
## k 7 -none- numeric   
## method 1 -none- character

plot(cv\_tree$size, cv\_tree$dev, type = 'p')



#The Standard Dev of 3 leafs seems to be the lowest from the Size to SDev graph above. Lets prune the tree to have 3 leaves.  
  
tree\_prune3 <- prune.tree(tree\_model, best = 3)  
plot(tree\_prune3)  
text(tree\_prune3)  
title('Pruned Tree with 3 leaves')



summary(tree\_prune3)

##   
## Regression tree:  
## snip.tree(tree = tree\_model, nodes = c(6L, 2L, 7L))  
## Variables actually used in tree construction:  
## [1] "Po1" "NW"   
## Number of terminal nodes: 3   
## Residual mean deviance: 76460 = 3364000 / 44   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -550.9 -181.8 -37.9 0.0 158.9 688.1

The ‘Residual mean deviance’ for 3 terminal nodes is 76460. This is much higher than the tree with 7 nodes, but the previous model may be overfitted. Lets predict the Crime values using the pruned tree. Once we have predictions, lets hand calculate a R2 squared value to see how great our pruned tree model is.

tree\_prune3.predict <- predict(tree\_prune3, data = data2[,1:15])  
  
tree\_prune3.predict

## 1 2 3 4 5 6 7 8   
## 669.6087 1304.9286 669.6087 1304.9286 886.9000 886.9000 1304.9286 1304.9286   
## 9 10 11 12 13 14 15 16   
## 669.6087 669.6087 1304.9286 669.6087 669.6087 669.6087 669.6087 1304.9286   
## 17 18 19 20 21 22 23 24   
## 669.6087 1304.9286 886.9000 1304.9286 669.6087 669.6087 1304.9286 886.9000   
## 25 26 27 28 29 30 31 32   
## 669.6087 1304.9286 669.6087 1304.9286 1304.9286 669.6087 669.6087 1304.9286   
## 33 34 35 36 37 38 39 40   
## 669.6087 886.9000 886.9000 886.9000 669.6087 669.6087 669.6087 1304.9286   
## 41 42 43 44 45 46 47   
## 669.6087 669.6087 669.6087 886.9000 669.6087 886.9000 886.9000

#In order to find the R2 value, we need to use a formula:  
# 1 - (Residual Sum of Squares)/(Total Sum of Squares)  
#   
RSS = sum((tree\_prune3.predict - data2[,16])^2)  
TSS = sum((data2[,16] - mean(data2[,16]))^2)  
R2 = 1 - RSS/TSS  
R2

## [1] 0.5111061

Our R2 is 0.511, which isn’t great. We can visually see that most of the predictions don’t really make much sense as with 3 leafs nodes, we can only have a very small number of possible results. Although we thought having 7 leaves resulting in overfitting, it seems to be better fitting than our pruned model. Next, lets try to create a Random Forest model.

rf = randomForest(Crime ~., data = data2)  
rf.predict <- predict(rf, data = data2[,1:15])  
RSS\_rf = sum((rf.predict - data2[,16])^2)  
R2\_rf = 1 - RSS\_rf/TSS  
R2\_rf

## [1] 0.4065048

The R2 for our Random Forest is .4065. This is worse than our pruned tree model. Lets try to create another Random Forest model with different parameters.

#using the predictors from the pruned tree, and adding Wealth.   
rf\_tuned = randomForest(Crime ~ Wealth + Po1 + NW, data = data2)  
rf\_tuned.predict <- predict(rf\_tuned, data = data2[,1:15])  
RSS\_rf\_tuned = sum((rf\_tuned.predict - data2[,16])^2)  
R2\_rf\_tuned = 1 - RSS\_rf\_tuned/TSS  
R2\_rf\_tuned

## [1] 0.4407773

#Correlation between Wealth and Po1 predictors  
cor(data2[,12],data2[,16])

## [1] 0.4413199

While there could be many combinations of parameters that could yield a better model, I decided to take 3 predictors: Wealth, Po1 and NW. The pruned tree model had created a tree using Po1 and Nw, and I added Wealth because I initally assumed, prior to delving into the data that Wealth (median value of transferable assets or family income) would be heavily correlated with the amount of Crime with an area. By using these 3 predictors, the RandomForest tuned model returns an R2 value of 0.42, which is slightly better than the R2\_rf model. However, it still isn’t a great model to use.

## Question 10.2

**Describe a situation or problem from your job, everyday life, current events, etc., for which a logistic regression model would be appropriate. List some (up to 5) predictors that you might use.**

A logistic regression model is great if the prediction needs be a binary or a probabilistic result. In my previous job, I worked at hedgefund that owned a portfolio of terms loans of publicly traded companies. Many of these loans were high yield or distressed, so during times of recessions, many of these positions could default on their payments or the loan itself. We could have used a logistic regression model to help determine the probability of default. If we could obtain history on debt payments of companies with similar size and industry, we could see what factor are present during scheduled payments and defaults. some factors we could have used are: Cash/Debt Ratio, Annual Free Cash Flow, and Public Credit Rating.

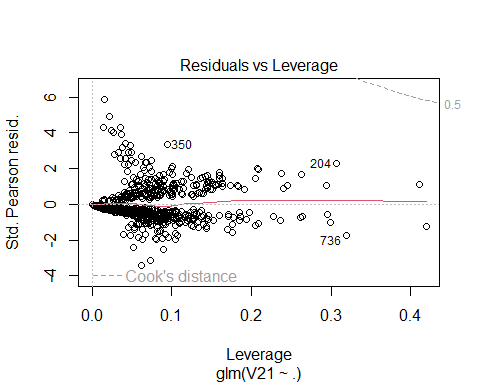
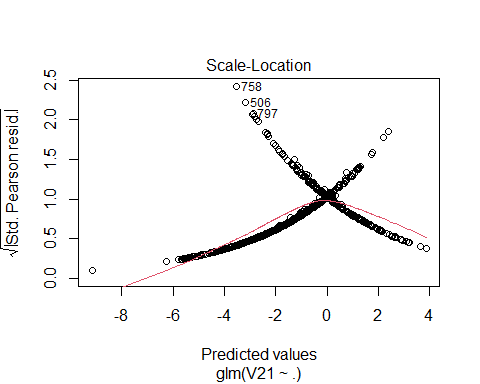
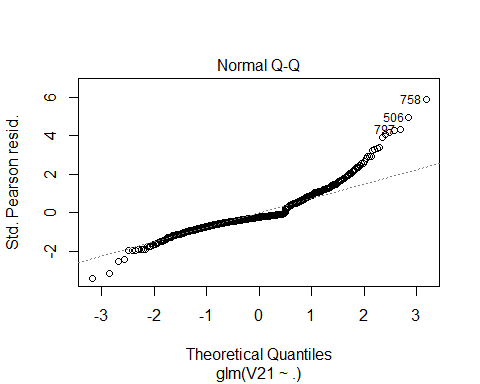
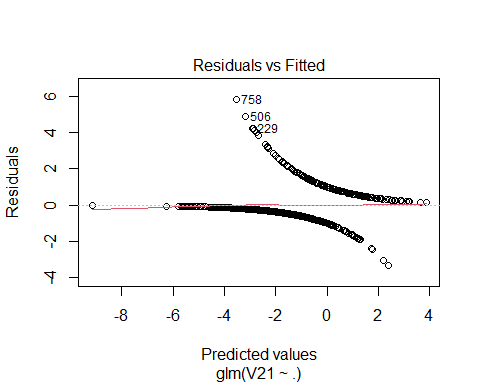
**Question 10.3**

1. **Using the GermanCredit data set germancredit.txt from** [**http://archive.ics.uci.edu/ml/machine-**](http://archive.ics.uci.edu/ml/machine-) **learning-databases/statlog/german / (description at** [**http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29**](http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29) **), use logistic regression to find a good predictive model for whether credit applicants are good credit risks or not. Show your model (factors used and their coefficients), the software output, and the quality of fit. You can use the glm function in R. To get a logistic regression (logit) model on data where the response is either zero or one, use family=binomial(link=”logit”) in your glm function call.**
2. **Because the model gives a result between 0 and 1, it requires setting a threshold probability to separate between “good” and “bad” answers. In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad. Determine a good threshold probability based on your model.**

data\_credit <- read.delim('germancredit.txt', header = FALSE, sep = ' ')  
head(data\_credit)

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18  
## 1 A11 6 A34 A43 1169 A65 A75 4 A93 A101 4 A121 67 A143 A152 2 A173 1  
## 2 A12 48 A32 A43 5951 A61 A73 2 A92 A101 2 A121 22 A143 A152 1 A173 1  
## 3 A14 12 A34 A46 2096 A61 A74 2 A93 A101 3 A121 49 A143 A152 1 A172 2  
## 4 A11 42 A32 A42 7882 A61 A74 2 A93 A103 4 A122 45 A143 A153 1 A173 2  
## 5 A11 24 A33 A40 4870 A61 A73 3 A93 A101 4 A124 53 A143 A153 2 A173 2  
## 6 A14 36 A32 A46 9055 A65 A73 2 A93 A101 4 A124 35 A143 A153 1 A172 2  
## V19 V20 V21  
## 1 A192 A201 1  
## 2 A191 A201 2  
## 3 A191 A201 1  
## 4 A191 A201 1  
## 5 A191 A201 2  
## 6 A192 A201 1

#replace 1 and 2 with 0 and 1 for V21  
data\_credit$V21[data\_credit$V21==1] <- 0  
data\_credit$V21[data\_credit$V21==2] <- 1  
  
#Lets split the data into train and test  
data\_credit <- data\_credit[complete.cases(data\_credit),]  
dt = sort(sample(nrow(data\_credit), nrow(data\_credit)\*.7))  
train<-data\_credit[dt,]  
test<-data\_credit[-dt,]  
  
#Create the model and show its summary  
log\_model <- glm(V21 ~., data = train, family = binomial(link = 'logit'))  
plot(log\_model)



summary(log\_model)

##   
## Call:  
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.2250 -0.6723 -0.3153 0.6222 2.6639   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -5.260e-02 1.486e+00 -0.035 0.971761   
## V1A12 -3.349e-01 2.698e-01 -1.241 0.214461   
## V1A13 -8.930e-01 4.416e-01 -2.022 0.043164 \*   
## V1A14 -1.697e+00 2.913e-01 -5.825 5.71e-09 \*\*\*  
## V2 3.681e-02 1.179e-02 3.122 0.001795 \*\*   
## V3A31 3.959e-01 7.319e-01 0.541 0.588536   
## V3A32 -7.895e-01 5.806e-01 -1.360 0.173905   
## V3A33 -1.353e+00 6.171e-01 -2.193 0.028329 \*   
## V3A34 -1.522e+00 5.910e-01 -2.576 0.010004 \*   
## V4A41 -2.140e+00 4.877e-01 -4.388 1.14e-05 \*\*\*  
## V4A410 -2.078e+00 1.060e+00 -1.960 0.050013 .   
## V4A42 -1.183e+00 3.303e-01 -3.582 0.000341 \*\*\*  
## V4A43 -1.002e+00 3.050e-01 -3.284 0.001024 \*\*   
## V4A44 -1.001e+00 8.909e-01 -1.124 0.260981   
## V4A45 -9.619e-01 7.260e-01 -1.325 0.185189   
## V4A46 -2.516e-01 5.289e-01 -0.476 0.634229   
## V4A48 -2.339e+00 1.298e+00 -1.801 0.071681 .   
## V4A49 -9.621e-01 4.345e-01 -2.214 0.026814 \*   
## V5 1.571e-04 5.547e-05 2.833 0.004617 \*\*   
## V6A62 -3.578e-01 3.480e-01 -1.028 0.303876   
## V6A63 -2.176e-01 5.003e-01 -0.435 0.663574   
## V6A64 -2.362e+00 8.198e-01 -2.882 0.003958 \*\*   
## V6A65 -1.214e+00 3.274e-01 -3.709 0.000208 \*\*\*  
## V7A72 1.062e-02 5.327e-01 0.020 0.984091   
## V7A73 -5.966e-01 5.129e-01 -1.163 0.244809   
## V7A74 -1.337e+00 5.699e-01 -2.346 0.018965 \*   
## V7A75 -6.797e-01 5.215e-01 -1.303 0.192408   
## V8 3.617e-01 1.104e-01 3.277 0.001048 \*\*   
## V9A92 -4.371e-01 5.287e-01 -0.827 0.408428   
## V9A93 -9.724e-01 5.124e-01 -1.898 0.057715 .   
## V9A94 -4.816e-01 6.017e-01 -0.800 0.423427   
## V10A102 7.567e-01 4.898e-01 1.545 0.122335   
## V10A103 -1.214e+00 5.125e-01 -2.369 0.017813 \*   
## V11 4.969e-02 1.078e-01 0.461 0.644870   
## V12A122 5.239e-01 3.118e-01 1.680 0.092968 .   
## V12A123 3.373e-01 2.944e-01 1.146 0.251986   
## V12A124 6.847e-01 5.447e-01 1.257 0.208741   
## V13 -2.313e-02 1.187e-02 -1.948 0.051466 .   
## V14A142 -2.550e-01 5.149e-01 -0.495 0.620381   
## V14A143 -5.385e-01 3.084e-01 -1.746 0.080735 .   
## V15A152 -4.466e-01 2.940e-01 -1.519 0.128675   
## V15A153 -3.411e-01 6.084e-01 -0.561 0.574980   
## V16 1.959e-01 2.338e-01 0.838 0.402025   
## V17A172 1.542e+00 8.921e-01 1.728 0.083966 .   
## V17A173 1.315e+00 8.633e-01 1.523 0.127715   
## V17A174 1.027e+00 8.801e-01 1.167 0.243366   
## V18 4.948e-01 3.096e-01 1.598 0.109950   
## V19A192 -4.070e-01 2.561e-01 -1.589 0.112062   
## V20A202 -1.934e+00 8.944e-01 -2.162 0.030584 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 866.74 on 699 degrees of freedom  
## Residual deviance: 596.69 on 651 degrees of freedom  
## AIC: 694.69  
##   
## Number of Fisher Scoring iterations: 5

log\_predict <- predict(log\_model, newdata = test[,1:20], type = 'response')  
  
log\_conMat <- confusionMatrix(reference = as.factor(test$V21), data = as.factor(round(log\_predict)))  
log\_conMat

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 189 44  
## 1 28 39  
##   
## Accuracy : 0.76   
## 95% CI : (0.7076, 0.8072)  
## No Information Rate : 0.7233   
## P-Value [Acc > NIR] : 0.08629   
##   
## Kappa : 0.3624   
##   
## Mcnemar's Test P-Value : 0.07710   
##   
## Sensitivity : 0.8710   
## Specificity : 0.4699   
## Pos Pred Value : 0.8112   
## Neg Pred Value : 0.5821   
## Prevalence : 0.7233   
## Detection Rate : 0.6300   
## Detection Prevalence : 0.7767   
## Balanced Accuracy : 0.6704   
##   
## 'Positive' Class : 0   
##

#Threshold Value = .5  
  
score = 0\*(log\_conMat$table[1]/300)+5\*(log\_conMat$table[2]/300)+1\*(log\_conMat$table[3]/300)+0\*(log\_conMat$table[4]/300)  
score

## [1] 0.6133333

Here are some of the results of from the Confusion Matrix: The Accuracy of the Model is .7533

Sensitivity : 0.7731 (TP/(TP+FN))  
Specificity : 0.6774 (TN/(TN+FP)) 0&0 –> 184 = True Positive(TP) 0&1 –> 54 = False Positive(FP) 1&0 –> 20 = False Negative(FN) 0&0 –> 42 = True Negative(TN)

However, it is 5x worse to classifying a bad customer as good than classifying a good customer as bad. We need to adjust the threshold value to reduce the number of bad customers classified as good (False Positive). Currently, we are rounding the values, any prediction with a .5 value is considered to be classified as good. We should make the good classification more strict - lets use .75.

In order to compare the current threshold output with the new threshold output, lets give the current Confusion Matrix a score. The confusion matrix with a threshold of .5 has a score of 0.513. The lower the score, the better.

threshold <- 0.75  
t2 <- as.matrix(table(round(log\_predict > .75), test$V21))  
t2

##   
## 0 1  
## 0 210 63  
## 1 7 20

score2 = 0\*(t2[1]/300)+5\*(t2[2]/300)+1\*(t2[3]/300)+0\*(t2[4]/300)  
score2

## [1] 0.3266667

With a threshold of .75, the score decrease to 0.283. Also noticeably, the False Positive only has 1 occurance out of 300.