# Daniel Rauscher

## Module 3 Assignment 2

parole <- read\_csv("parole.csv")

##   
## ── Column specification ────────────────────────────────────────────────────────  
## cols(  
## male = col\_double(),  
## race = col\_double(),  
## age = col\_double(),  
## state = col\_double(),  
## time.served = col\_double(),  
## max.sentence = col\_double(),  
## multiple.offenses = col\_double(),  
## crime = col\_double(),  
## violator = col\_double()  
## )

view(parole)

parole = parole %>% mutate(male = as\_factor(male)) %>% mutate(race = as\_factor(race)) %>% mutate(state = as\_factor(state)) %>% mutate(crime = as\_factor(crime)) %>% mutate(multiple.offenses = as\_factor(multiple.offenses)) %>% mutate(violator = as\_factor(violator)) %>%  
 mutate(male = fct\_recode(male, "No" = "0", "Yes" = "1" )) %>%   
 mutate(race = fct\_recode(race, "Other" = "2", "White" = "1" )) %>%   
 mutate(state = fct\_recode(state, "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4", "Other" = "1" )) %>%   
 mutate(crime = fct\_recode(crime, "Larceny" = "2", "Drug-related" = "3", "Driving-related" = "4", "Other" = "1" )) %>%   
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "No" = "0", "Yes" = "1" )) %>%  
 mutate(violator = fct\_recode(violator, "No" = "0", "Yes" = "1" ))  
  
view(parole)

## Task 1

set.seed(12345)  
parole\_split <- initial\_split(parole, prob = 0.70, strata = violator)  
train = training(parole\_split)  
test = testing(parole\_split)

## Task 2

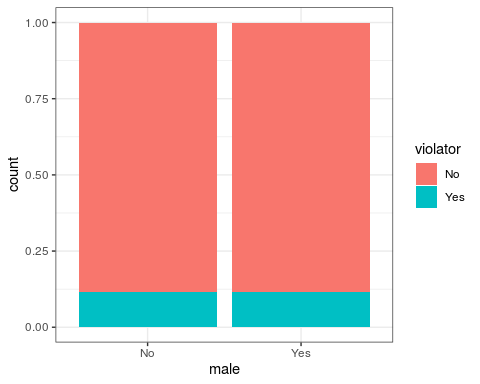
The strongest predictors of being a violator are the State, max sentence, and multiple offenses.

State variable seems to strongly predict violators with being in Virginia predicting low violators and Louisiana being the highest. Would need more information about each states parole rules/guidance to understand why this makes sense.

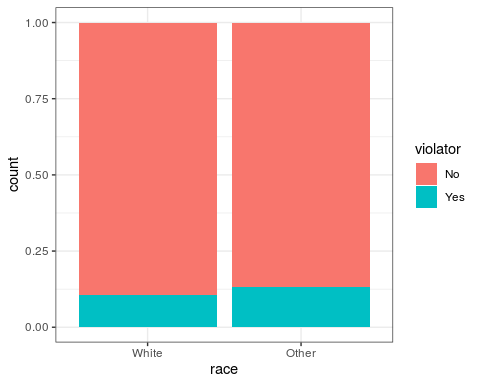
Max Sentence does seem to be a strong predictor of being a violator. The median max sentence is lower for being a violator.

Multiple offenses also seems to be a strong predictor of being a violator. Having multiple offenses raises the chances of being a violator by almost 6%.

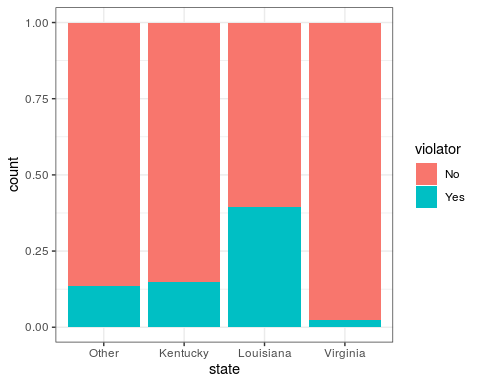
ggplot(train, aes(x=male, fill=violator)) + geom\_bar(position = "fill") + theme\_bw()



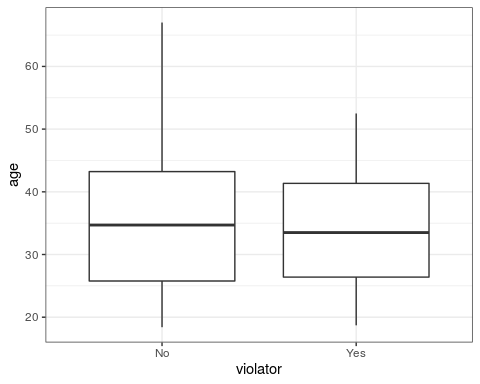
ggplot(train, aes(x=race, fill=violator)) + geom\_bar(position = "fill") + theme\_bw()



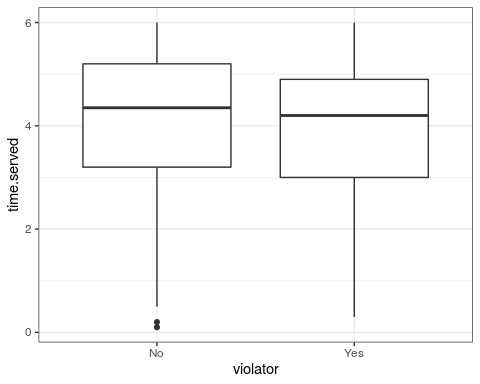
ggplot(train, aes(x=state, fill=violator)) + geom\_bar(position = "fill") + theme\_bw()



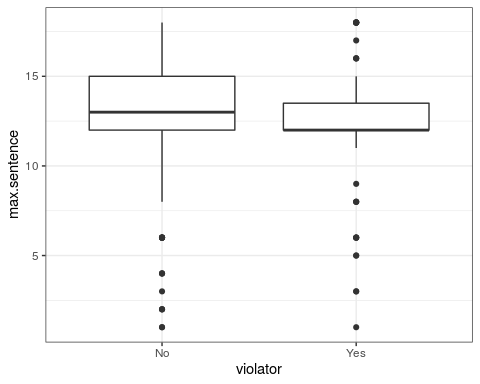
ggplot(train, aes(x=violator, y=age)) + geom\_boxplot() + theme\_bw()



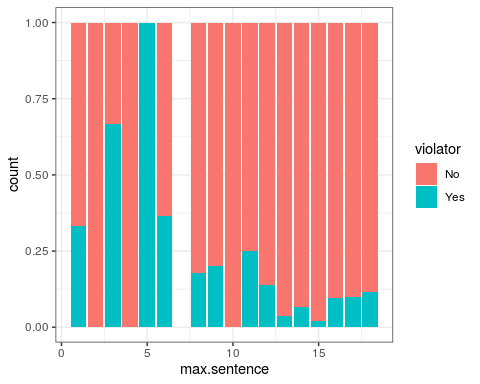
ggplot(train, aes(x=violator, y=time.served)) + geom\_boxplot() + theme\_bw()



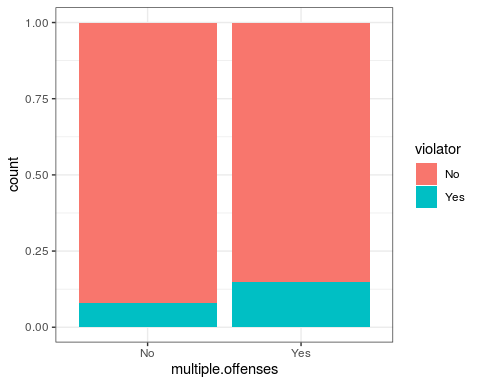
ggplot(train, aes(x=violator, y=max.sentence)) + geom\_boxplot() + theme\_bw()



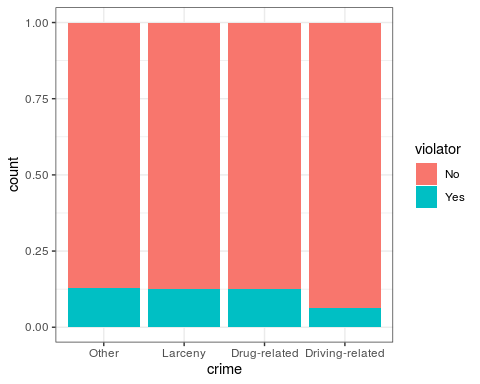
ggplot(train, aes(x=max.sentence, fill=violator)) + geom\_bar(position = "fill") + theme\_bw()



ggplot(train, aes(x=multiple.offenses, fill=violator)) + geom\_bar(position = "fill") + theme\_bw()



ggplot(train, aes(x=crime, fill=violator)) + geom\_bar(position = "fill") + theme\_bw()



t7 = table(train$violator, train$race)  
prop.table(t7, margin =2)

##   
## White Other  
## No 0.8951049 0.8687783  
## Yes 0.1048951 0.1312217

t8 = table(train$violator, train$multiple.offenses)  
prop.table(t8, margin =2)

##   
## No Yes  
## No 0.91983122 0.85185185  
## Yes 0.08016878 0.14814815

t9 = table(train$violator, train$crime)  
prop.table(t9, margin =2)

##   
## Other Larceny Drug-related Driving-related  
## No 0.87280702 0.87356322 0.87610619 0.93670886  
## Yes 0.12719298 0.12643678 0.12389381 0.06329114

## Task 3

The model suggests that multiple offenses is significant based on the P-value. Since we do not have an AIC to compare it too, we do not know much by looking at it alone. By looking at the coefficients we can see that by having multiple offenses one is more likely to be a violator than if they do not have multiple offenses.

parole\_model =   
 logistic\_reg(mode = "classification") %>%   
 set\_engine("glm")   
  
parole\_recipe = recipe(violator ~ multiple.offenses,train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe) %>%   
 add\_model(parole\_model)  
  
parole\_fit = fit(logreg\_wf, train)

summary(parole\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.5663 -0.5663 -0.4088 -0.4088 2.2466   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.4401 0.2392 -10.201 <2e-16 \*\*\*  
## multiple.offenses\_Yes 0.6909 0.2942 2.348 0.0189 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 364.67 on 506 degrees of freedom  
## Residual deviance: 358.85 on 505 degrees of freedom  
## AIC: 362.85  
##   
## Number of Fisher Scoring iterations: 5

## Task 4

The first model I built, I threw all variables into the model. By looking at the model I can see which are significant and which ones are not. The AIC did improve from the Task 3 model, so this first model did get better.

The second model I built, I removed all the insignificant variables, only leaving State and multiple offenses variables to predict violator. This model lowered the AIC even more than the 2nd model built.

parole\_model2 =   
 logistic\_reg(mode = "classification") %>%   
 set\_engine("glm")   
  
parole\_recipe2 = recipe(violator ~ multiple.offenses + male + race + age + state + time.served + max.sentence + crime ,train) %>%   
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe2) %>%   
 add\_model(parole\_model2)  
  
parole\_fit2 = fit(logreg\_wf, train)

summary(parole\_fit2$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.5189 -0.4324 -0.2695 -0.1793 2.8067   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.039052 1.116175 -2.723 0.00647 \*\*   
## age 0.004903 0.015346 0.320 0.74933   
## time.served -0.079212 0.113591 -0.697 0.48559   
## max.sentence 0.039556 0.048019 0.824 0.41008   
## multiple.offenses\_Yes 1.590077 0.372029 4.274 1.92e-05 \*\*\*  
## male\_Yes 0.281802 0.439946 0.641 0.52182   
## race\_Other 0.443687 0.358593 1.237 0.21598   
## state\_Kentucky 0.544529 0.480560 1.133 0.25717   
## state\_Louisiana 0.774125 0.508455 1.523 0.12788   
## state\_Virginia -2.948822 0.576468 -5.115 3.13e-07 \*\*\*  
## crime\_Larceny 0.316021 0.473564 0.667 0.50456   
## crime\_Drug.related -0.394782 0.410285 -0.962 0.33594   
## crime\_Driving.related -0.311261 0.562188 -0.554 0.57981   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 364.67 on 506 degrees of freedom  
## Residual deviance: 274.08 on 494 degrees of freedom  
## AIC: 300.08  
##   
## Number of Fisher Scoring iterations: 6

parole\_model3 =   
 logistic\_reg(mode = "classification") %>%   
 set\_engine("glm")   
  
parole\_recipe3 = recipe(violator ~ multiple.offenses +state ,train) %>%   
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe3) %>%   
 add\_model(parole\_model3)  
  
parole\_fit3 = fit(logreg\_wf, train)

summary(parole\_fit3$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.2000 -0.4952 -0.2460 -0.2460 2.6505   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.4971 0.3565 -7.005 2.47e-12 \*\*\*  
## multiple.offenses\_Yes 1.6319 0.3663 4.456 8.37e-06 \*\*\*  
## state\_Kentucky 0.4601 0.4451 1.034 0.3013   
## state\_Louisiana 0.9181 0.4114 2.231 0.0257 \*   
## state\_Virginia -2.6172 0.5332 -4.908 9.20e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 364.67 on 506 degrees of freedom  
## Residual deviance: 279.41 on 502 degrees of freedom  
## AIC: 289.41  
##   
## Number of Fisher Scoring iterations: 6

## Task 5

In this model it appears the variables State, and multiple offenses are significant, while race is not significant. The AIC is very similar to my parole\_model3 above which only contained the variables state and multiple offenses. I did notice that adding in race, changed State\_Louisiana from being significant to insignificant in this model.

parole\_model4 =   
 logistic\_reg(mode = "classification") %>%   
 set\_engine("glm")   
  
parole\_recipe4 = recipe(violator ~ multiple.offenses +state + race ,train) %>%   
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe4) %>%   
 add\_model(parole\_model4)  
  
parole\_fit4 = fit(logreg\_wf, train)

summary(parole\_fit4$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.2598 -0.4718 -0.2675 -0.2173 2.7414   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.5431 0.3579 -7.106 1.20e-12 \*\*\*  
## multiple.offenses\_Yes 1.5998 0.3684 4.342 1.41e-05 \*\*\*  
## state\_Kentucky 0.4036 0.4470 0.903 0.367   
## state\_Louisiana 0.7135 0.4481 1.592 0.111   
## state\_Virginia -2.7907 0.5570 -5.010 5.43e-07 \*\*\*  
## race\_Other 0.4215 0.3527 1.195 0.232   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 364.67 on 506 degrees of freedom  
## Residual deviance: 277.99 on 501 degrees of freedom  
## AIC: 289.99  
##   
## Number of Fisher Scoring iterations: 6

## Task 6

According to our model, Parolee 1 is more likely to be a violator than Parolee 2.

Parolee1 = data.frame(state = "Louisiana", multiple.offenses = "Yes", race = "White")  
predict(parole\_fit4, Parolee1, type="prob")

## # A tibble: 1 x 2  
## .pred\_No .pred\_Yes  
## <dbl> <dbl>  
## 1 0.557 0.443

Parolee2 = data.frame(state = "Kentucky", multiple.offenses = "No", race = "Other")  
predict(parole\_fit4, Parolee2, type="prob")

## # A tibble: 1 x 2  
## .pred\_No .pred\_Yes  
## <dbl> <dbl>  
## 1 0.848 0.152

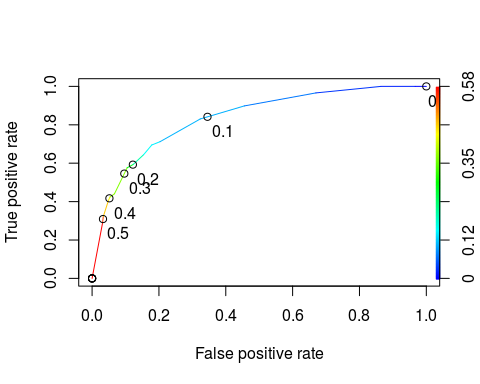
## Task 7

The best probability threshold that best balances specificity and sensitivity is 0.107.

predictions = predict(parole\_fit4, train, type="prob")[2]  
head(predictions)

## # A tibble: 6 x 1  
## .pred\_Yes  
## <dbl>  
## 1 0.0729  
## 2 0.0729  
## 3 0.0729  
## 4 0.107   
## 5 0.107   
## 6 0.0729

ROCRpred = prediction(predictions, train$violator)  
  
ROCRperf = performance(ROCRpred, "tpr", "fpr")  
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))



as.numeric(performance(ROCRpred, "auc")@y.values)

## [1] 0.834916

opt.cut = function(perf, pred){  
 cut.ind = mapply(FUN=function(x, y, p){  
 d = (x - 0)^2 + (y-1)^2  
 ind = which(d == min(d))  
 c(sensitivity = y[[ind]], specificity = 1-x[[ind]],   
 cutoff = p[[ind]])  
 }, perf@x.values, perf@y.values, pred@cutoffs)  
}  
print(opt.cut(ROCRperf, ROCRpred))

## [,1]  
## sensitivity 0.7118644  
## specificity 0.7968750  
## cutoff 0.1070172

## Task 8

Using a cutoff of 0.107, the sensitivity is .712 and specificity is 0.797. The accuracy at this cutoff is 0.807 or 80.7% accurate.

The implications of incorrectly classifying a parolee could lead to less oversight on that parolee letting them violate their parole easier. Every parolee is likely still being made to abide by their parole terms so incorrectly classifying a parolee is a surprise to the parole officer, but no real threat comes from it.

t1 = table(train$violator,predictions > 0.1070172)  
t1

##   
## FALSE TRUE  
## No 368 80  
## Yes 18 41

(t1[1,1]+t1[2,2])/nrow(train)

## [1] 0.8067061

## Task 9

The probability threshold that best maximizes accuracy on the training set is 0.47

t2 = table(train$violator,predictions > 0.47)  
t2

##   
## FALSE TRUE  
## No 433 15  
## Yes 40 19

(t2[1,1]+t2[2,2])/nrow(train)

## [1] 0.8915187

## Task 10

Using our model from the training set, we can predict an accuracy of 92.26% on our testing set.

predictions\_test = predict(parole\_fit4, test, type="prob")[2]  
head(predictions\_test)

## # A tibble: 6 x 1  
## .pred\_Yes  
## <dbl>  
## 1 0.107   
## 2 0.0729  
## 3 0.107   
## 4 0.0729  
## 5 0.0729  
## 6 0.0729

t3 = table(test$violator,predictions\_test > 0.47)  
t3

##   
## FALSE TRUE  
## No 148 1  
## Yes 12 7

(t3[1,1]+t3[2,2])/nrow(test)

## [1] 0.922619