# Module 3 Assignment 2

## Madison Rauscher

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()  
## )

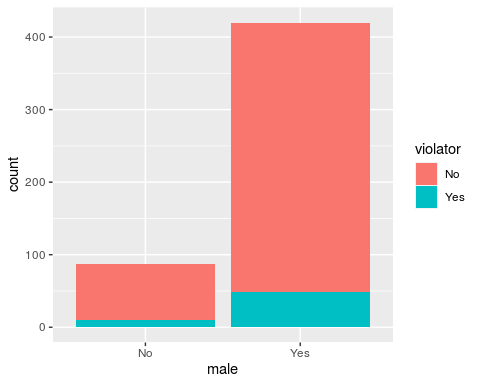
parole <- parole %>%  
 mutate(male=as\_factor(male)) %>%  
 mutate(race=as\_factor(race)) %>%  
 mutate(state=as\_factor(state)) %>%  
 mutate(crime=as\_factor(crime)) %>%  
 mutate(multiple.offenses=as\_factor(multiple.offenses)) %>%  
 mutate(violator=as\_factor(violator)) %>%  
 mutate(male = fct\_recode(male, "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, "Yes" = "1", "No" = "0")) %>%  
 mutate(violator = fct\_recode(violator, "Yes" = "1", "No" = "0"))

**Task 1**

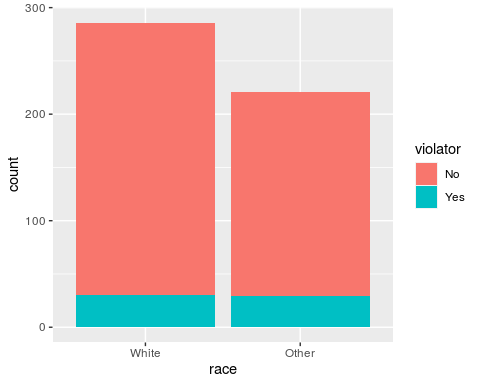
set.seed(12345)  
parole\_split= initial\_split(parole, prob = 0.70, strata=violator)  
train= training(parole\_split)  
test= testing(parole\_split)

**Task 2**

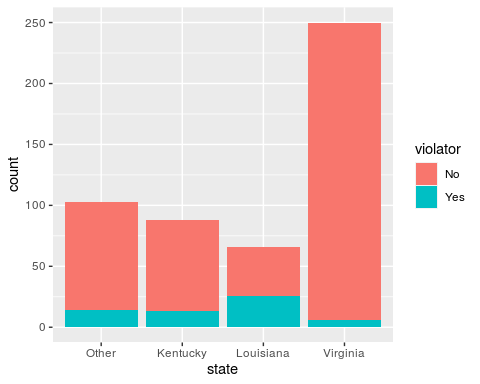
ggplot(train, aes(male, fill=violator))+ geom\_bar()



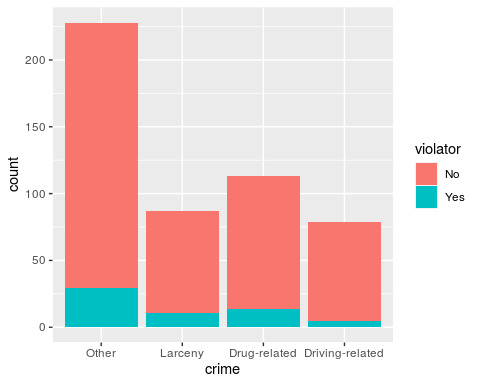
ggplot(train, aes(race, fill=violator))+ geom\_bar()



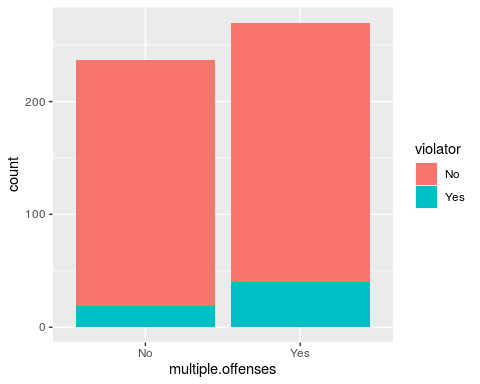
ggplot(train, aes(state, fill=violator))+ geom\_bar()



ggplot(train, aes(crime, fill=violator))+ geom\_bar()



ggplot(train, aes(multiple.offenses, fill=violator))+ geom\_bar()



t1= table(parole$violator, parole$male)  
prop.table(t1, margin=2)

##   
## No Yes  
## No 0.8923077 0.8825688  
## Yes 0.1076923 0.1174312

t2= table(parole$violator, parole$race)  
prop.table(t2, margin=2)

##   
## White Other  
## No 0.90488432 0.85664336  
## Yes 0.09511568 0.14335664

t3= table(parole$violator, parole$state)  
prop.table(t3, margin=2)

##   
## Other Kentucky Louisiana Virginia  
## No 0.86013986 0.88333333 0.54878049 0.97878788  
## Yes 0.13986014 0.11666667 0.45121951 0.02121212

t4= table(parole$violator, parole$crime)  
prop.table(t4, margin=2)

##   
## Other Larceny Drug-related Driving-related  
## No 0.87619048 0.87735849 0.87581699 0.93069307  
## Yes 0.12380952 0.12264151 0.12418301 0.06930693

t5= table(parole$violator, parole$multiple.offenses)  
prop.table(t5, margin=2)

##   
## No Yes  
## No 0.9201278 0.8535912  
## Yes 0.0798722 0.1464088

I used stacked bar charts to visualize the data in order to predict whether or not a parolee will violate his/her parole. Most charts shows very small differences so I went ahead and made tables for each for further clarification. At first glance of the chart, gender seems to be an predictor of “violator”, however there are many more males than females so it was necessary to consult the table which showed that there is actually only a very small difference between the two. Both the chart and table for race confirmed there is little difference and it is likely not a strong predictor either. State on the other hand is strongly predictive of “violator.” Those in Louisiana are very likely to violate while those in Virginia are very unlikely to. Crime is somewhat an indicator as those who committed driving-related crimes are much less likely to violate parole while the other three categories are equally as likely. Finally, multiple offenses seems like it will be a pretty good indicator as those who have committed multiple offenses are more likely to violate parole.

**Task 3**  
State appears to be the most predictive of “violator”.

parole\_model=  
 logistic\_reg(mode="classification") %>%  
 set\_engine("glm")  
  
parole\_recipe = recipe(violator ~ state, train)  
  
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   
## -1.0008 -0.5405 -0.2204 -0.2204 2.7312   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.84958 0.28751 -6.433 1.25e-10 \*\*\*  
## stateKentucky 0.09704 0.41584 0.233 0.815481   
## stateLouisiana 1.41880 0.38226 3.712 0.000206 \*\*\*  
## stateVirginia -1.85583 0.50341 -3.686 0.000227 \*\*\*  
## ---  
## 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: 300.70 on 503 degrees of freedom  
## AIC: 308.7  
##   
## Number of Fisher Scoring iterations: 6

All levels are highly significant except Kentucky which could be concerning, but still gives us reason to explore further. There is no R-squared in this model so we can look at AIC as an indicator of quality. It is 308.7 which seems high and an indicator of poor quality, but we will not know for sure until we are able to compare it going forward.

**Task 4**

parole\_model=  
 logistic\_reg(mode="classification") %>%  
 set\_engine("glm")  
  
parole\_recipe = recipe(violator ~ state + multiple.offenses, train)  
  
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   
## -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 \*\*\*  
## stateKentucky 0.4601 0.4451 1.034 0.3013   
## stateLouisiana 0.9181 0.4114 2.231 0.0257 \*   
## stateVirginia -2.6172 0.5332 -4.908 9.20e-07 \*\*\*  
## multiple.offensesYes 1.6319 0.3663 4.456 8.37e-06 \*\*\*  
## ---  
## 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

The best model I could find to predict “violator” was using the variables state and multiple offenses. The AIC is the lowest of any combination at 289.41. The sate of Virginia and having multiple offenses are the most significant while the state of Louisiana is also significant. The model does seem intuitive because it does make sense that someone with multiple offenses would be more willing to risk breaking parole as they might be frustrated or tired of being controlled. It also is intuitive regarding state because Virginia likely has more funding to maintain stricter parole protocols and offender may have a harder time breaking parole or feel more intimidated not to.

**Task 5**

parole\_model=  
 logistic\_reg(mode="classification") %>%  
 set\_engine("glm")  
  
parole\_recipe = recipe(violator ~ state + multiple.offenses + race, train)  
  
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   
## -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 \*\*\*  
## stateKentucky 0.4036 0.4470 0.903 0.367   
## stateLouisiana 0.7135 0.4481 1.592 0.111   
## stateVirginia -2.7907 0.5570 -5.010 5.43e-07 \*\*\*  
## multiple.offensesYes 1.5998 0.3684 4.342 1.41e-05 \*\*\*  
## raceOther 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

Based on the 289.99 AIC this model is almost as high of quality as the previous model I created in take 4. Adding race only decreases the quality slightly. The state of Virginia and having multiple offenses are still significant while Louisiana is no longer significant. Race is not significant.

**Task 6**

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

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

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

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

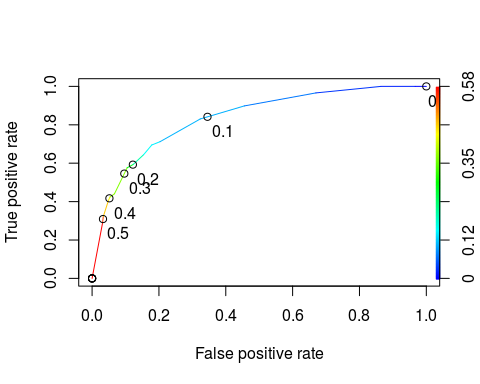
The predicted probability of parole violation for Parolee1 is .44 and for Parolee2 it is .15.

**Task 7**

predictions= predict(parole\_fit, 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(-.2,1.7))



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

The best probability threshold that best balances specificity and sensitivity on the training set is .11 when rounded to two decimal places.

**Task 8**

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

The accuracy is 80.67%, the sensitivity pulled from task 7 is .71, the specificity pulled from task 7 is .80, rounded. The implications of incorrectly classifying a parolee is inefficiently allocating resources based on risk and the potential that someone may commit another crime if not watched closely due to the fact that they were incorrectly classified.

**Task 9**

t1 = table(train$violator,predictions > 0.45)  
t1

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

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

## [1] 0.8915187

Using trial and error I found the probability threshold .45 best maximizes the accuracy on the training set at 89.15%.

**Task 10**

predictions\_test= predict(parole\_fit, 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

t2 = table(test$violator,predictions\_test > 0.45)  
t2

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

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

## [1] 0.922619

The accuracy of the model ont he testing set using the .45 probability threshold is 92.26%.