# Model Validation

## Katherine Schumann

### Mod 3 Assignment 1

parole <- read\_csv("C:/Users/kathe/OneDrive/Desktop/Ban 502/Module 3/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(male = fct\_recode(male, "Male" = "1", "Female" = "0"))  
  
  
parole = parole%>% mutate(race = as\_factor(race))%>% mutate(race = fct\_recode(race, "White" = "1", "Other" = "2"))  
  
parole = parole%>% mutate(state = as\_factor(state))%>% mutate(state = fct\_recode(state, "Other" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4"))  
  
parole = parole%>% mutate(multiple.offenses = as\_factor(multiple.offenses)) %>% mutate(multiple.offenses = fct\_recode(multiple.offenses, "Yes" = "1", "No" = "0"))  
  
parole = parole%>% mutate(crime = as\_factor(crime)) %>% mutate(crime = fct\_recode(crime, "Other" = "1", "Larceny" = "2", "Drug-related" = "3", "Driving-related" = "4"))  
  
parole = parole %>% mutate(violator = as.character.numeric\_version(violator)) %>% mutate(violator = fct\_recode(violator, "Yes" = "1", "No" = "0"))

summary(parole)

## male race age state time.served   
## Female:130 White:389 Min. :18.40 Other :143 Min. :0.000   
## Male :545 Other:286 1st Qu.:25.35 Kentucky :120 1st Qu.:3.250   
## Median :33.70 Louisiana: 82 Median :4.400   
## Mean :34.51 Virginia :330 Mean :4.198   
## 3rd Qu.:42.55 3rd Qu.:5.200   
## Max. :67.00 Max. :6.000   
## max.sentence multiple.offenses crime violator   
## Min. : 1.00 No :313 Other :315 No :597   
## 1st Qu.:12.00 Yes:362 Larceny :106 Yes: 78   
## Median :12.00 Drug-related :153   
## Mean :13.06 Driving-related:101   
## 3rd Qu.:15.00   
## Max. :18.00

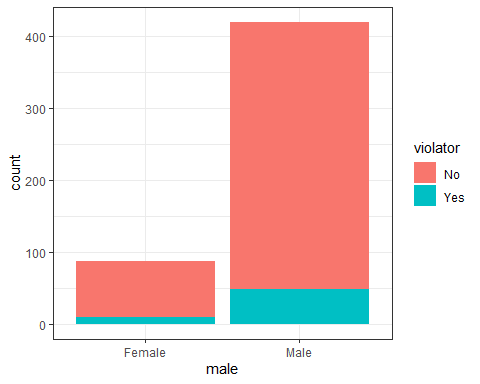
### Task 1

Split

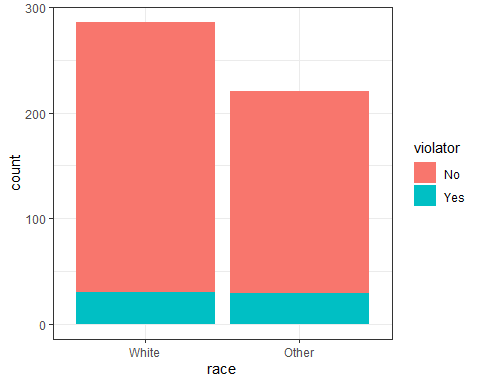
set.seed(12345)  
parole\_split = initial\_split(parole, prob = 0.70, strata = violator)  
train = training(parole\_split)  
test = testing(parole\_split)

### Task 2

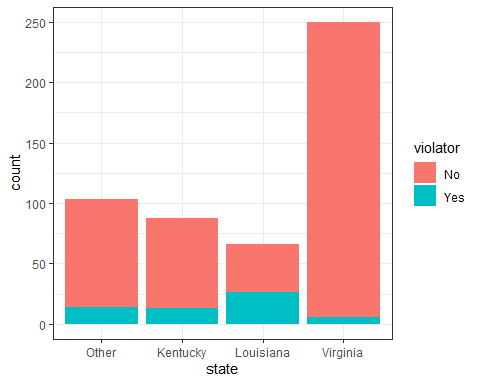
ggplot(train, aes(x=male, fill = violator)) + geom\_bar() + theme\_bw()



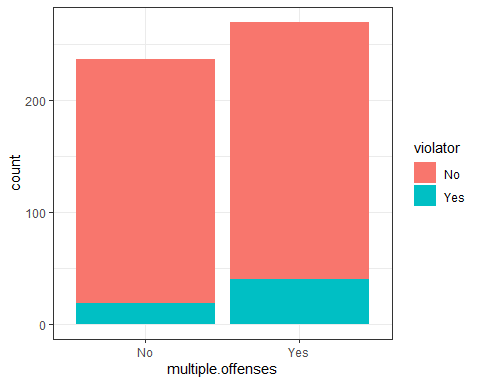
ggplot(train, aes(x=race, fill = violator)) + geom\_bar() + theme\_bw()



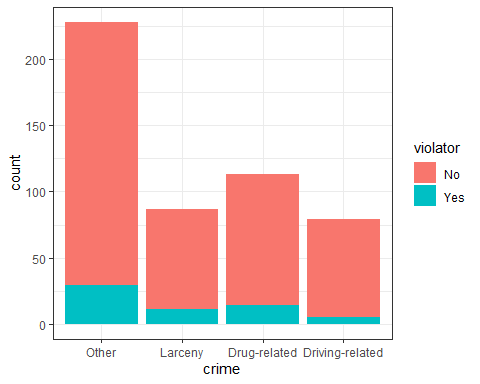
ggplot(train, aes(x=state, fill = violator)) + geom\_bar() + theme\_bw()



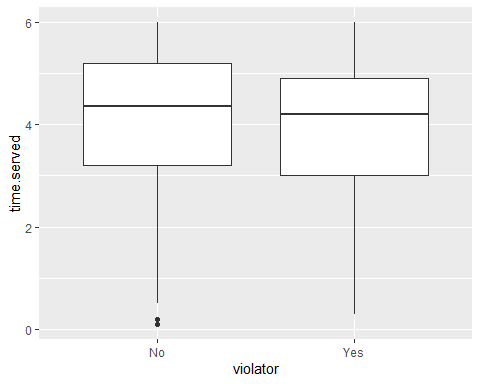
ggplot(train, aes(x=multiple.offenses, fill = violator)) + geom\_bar() + theme\_bw()



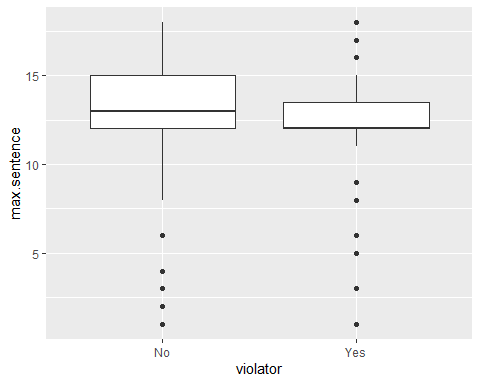
ggplot(train, aes(x=crime, fill = violator)) + geom\_bar() + theme\_bw()



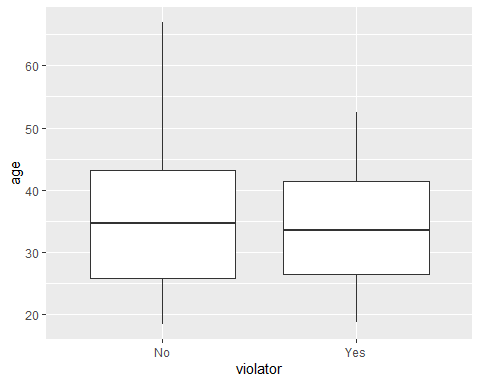
ggplot(train,aes(x=violator, y=time.served)) + geom\_boxplot()



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



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

 After observing the variables above I would suggest that age seems to be a fairly good predictor variable and so does state specifically for the state of Louisiana.

### Task 3

trainage\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
trainage\_recipe = recipe(violator ~ age, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(trainage\_recipe) %>%   
 add\_model(trainage\_model)  
  
trainage\_fit = fit(logreg\_wf, train)

summary(trainage\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.5303 -0.5128 -0.4938 -0.4743 2.1375   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.736273 0.487029 -3.565 0.000364 \*\*\*  
## age -0.008396 0.013606 -0.617 0.537148   
## ---  
## 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: 364.28 on 505 degrees of freedom  
## AIC: 368.28  
##   
## Number of Fisher Scoring iterations: 4

trainstate\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
trainstate\_recipe = recipe(violator ~ state, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(trainstate\_recipe) %>%   
 add\_model(trainstate\_model)  
  
trainstate\_fit = fit(logreg\_wf, train)

summary(trainstate\_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 \*\*\*  
## state\_Kentucky 0.09704 0.41584 0.233 0.815481   
## state\_Louisiana 1.41880 0.38226 3.712 0.000206 \*\*\*  
## state\_Virginia -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

State demonstrates to be a good fit according to the AIC value. I thought that the graphs demonstrated the biggest jump with state and age so I modeled those two, I think that state is a very good predictive model.

### Task 4

trainrace\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
trainrace\_recipe = recipe(violator ~ race, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(trainrace\_recipe) %>%   
 add\_model(trainrace\_model)  
  
trainrace\_fit = fit(logreg\_wf, train)

summary(trainrace\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.5304 -0.5304 -0.4708 -0.4708 2.1236   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.1440 0.1930 -11.111 <2e-16 \*\*\*  
## race\_Other 0.2538 0.2774 0.915 0.36   
## ---  
## 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: 363.83 on 505 degrees of freedom  
## AIC: 367.83  
##   
## Number of Fisher Scoring iterations: 4

trainmale\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
trainmale\_recipe = recipe(violator ~ male, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(trainmale\_recipe) %>%   
 add\_model(trainmale\_model)  
  
trainmale\_fit = fit(logreg\_wf, train)

summary(trainmale\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.4981 -0.4981 -0.4981 -0.4942 2.0801   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.04122 0.33613 -6.073 1.26e-09 \*\*\*  
## male\_Male 0.01684 0.36890 0.046 0.964   
## ---  
## 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: 364.66 on 505 degrees of freedom  
## AIC: 368.66  
##   
## Number of Fisher Scoring iterations: 4

traintimeserved\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
traintimeserved\_recipe = recipe(violator ~ time.served, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(traintimeserved\_recipe) %>%   
 add\_model(traintimeserved\_model)  
  
traintimeserved\_fit = fit(logreg\_wf, train)

summary(traintimeserved\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.7372 -0.5311 -0.4742 -0.4273 2.2536   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.1421 0.4382 -2.607 0.00915 \*\*  
## time.served -0.2192 0.1064 -2.061 0.03933 \*   
## ---  
## 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: 360.55 on 505 degrees of freedom  
## AIC: 364.55  
##   
## Number of Fisher Scoring iterations: 4

trainmax\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
trainmax\_recipe = recipe(violator ~ max.sentence, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(trainmax\_recipe) %>%   
 add\_model(trainmax\_model)  
  
traimax\_fit = fit(logreg\_wf, train)

summary(traimax\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.9763 -0.5150 -0.4838 -0.4002 2.3714   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.3607 0.5059 -0.713 0.475908   
## max.sentence -0.1327 0.0403 -3.294 0.000988 \*\*\*  
## ---  
## 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: 354.29 on 505 degrees of freedom  
## AIC: 358.29  
##   
## Number of Fisher Scoring iterations: 5

trainmulti\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
trainmulti\_recipe = recipe(violator ~ multiple.offenses, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(trainmulti\_recipe) %>%   
 add\_model(trainmulti\_model)  
  
trainmulti\_fit = fit(logreg\_wf, train)

summary(trainmulti\_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

traincrime\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
traincrime\_recipe = recipe(violator ~ crime, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(traincrime\_recipe) %>%   
 add\_model(traincrime\_model)  
  
traincrime\_fit = fit(logreg\_wf, train)

summary(traincrime\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.5216 -0.5216 -0.5200 -0.3616 2.3495   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.926009 0.198766 -9.690 <2e-16 \*\*\*  
## crime\_Larceny -0.006829 0.378913 -0.018 0.986   
## crime\_Drug.related -0.030054 0.347904 -0.086 0.931   
## crime\_Driving.related -0.768618 0.503010 -1.528 0.127   
## ---  
## 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: 361.73 on 503 degrees of freedom  
## AIC: 369.73  
##   
## Number of Fisher Scoring iterations: 5

The lowest AIC value comes from the state model, for the state of Virginia and Louisiana they are both significant variables. The other significant variable comes from the multiple offenses category. The quality of the state model is fairly good with an AIC value of 314.65. This model is not quite what I expected, I expected Louisiana to have a higher significance over Virginia.

### Task 5

train5\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
train5\_recipe = recipe(violator ~ state + multiple.offenses + race, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(train5\_recipe) %>%   
 add\_model(train5\_model)  
  
train5\_fit = fit(logreg\_wf, train)

summary(train5\_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 \*\*\*  
## 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 \*\*\*  
## multiple.offenses\_Yes 1.5998 0.3684 4.342 1.41e-05 \*\*\*  
## 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

The quality of this model is better than just the state model according to the AIC value since it is lower. The State (particularly Virginia) and the multiple offenses are both significant variables.

### Task 6

Parolee1 = data.frame(state = "Louisiana", multiple.offenses = "Yes", race = "White")  
predict(train5\_fit, 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(train5\_fit, Parolee2, 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 Parolee 1 is 41.24% and the predicted probability of parole violation for Parolee 2 is 13.9.5%.

### Task 7

Develop predictions

predictions = predict(train5\_fit, train, type="prob") #develop predicted probabilities  
head(predictions)

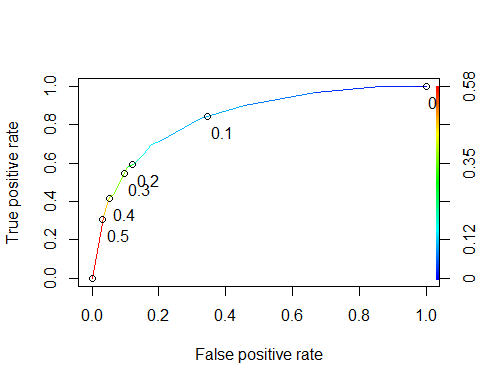
## # A tibble: 6 x 2  
## .pred\_No .pred\_Yes  
## <dbl> <dbl>  
## 1 0.927 0.0729  
## 2 0.927 0.0729  
## 3 0.927 0.0729  
## 4 0.893 0.107   
## 5 0.893 0.107   
## 6 0.927 0.0729

predictions = predict(train5\_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

Threshold selection

#Change this next line to the names of your predictions and the response variable in the training data frame  
ROCRpred = prediction(predictions, train$violator)  
  
###You shouldn't need to ever change the next two lines:  
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

### Task 8

#Determine threshold to balance sensitivity and specificity  
#DO NOT modify this code  
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

Test thresholds to evaluate accuracy

#confusion matrix  
#The "No" and "Yes" represent the actual values  
#The "FALSE" and "TRUE" represent our predicted values  
t1 = table(train$violator,predictions > 0.1371209)  
t1

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

Calculate accuracy

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

## [1] 0.8067061

Sensitivity

39/(20+39)

## [1] 0.6610169

Specificity

374/(374+74)

## [1] 0.8348214

The implications of incorrectly classifying a parolee would potentially mean that if we are trying to find a bail average we may state that they should stay in the jail rather than get parole.

### Task 9

Threshold = 0.6

t1 = table(train$violator,predictions > 0.5)  
t1

##   
## FALSE TRUE  
## No 434 14  
## Yes 42 17

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

## [1] 0.8895464

I think that this is the highest accuracy that we can get with the training set.

### Task 10

predictions = predict(train5\_fit, test, type="prob") #develop predicted probabilities  
head(predictions)

## # A tibble: 6 x 2  
## .pred\_No .pred\_Yes  
## <dbl> <dbl>  
## 1 0.893 0.107   
## 2 0.927 0.0729  
## 3 0.893 0.107   
## 4 0.927 0.0729  
## 5 0.927 0.0729  
## 6 0.927 0.0729

predictions = predict(train5\_fit, test, type="prob")[2]  
head(predictions)

## # 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

Test thresholds to evaluate accuracy

#confusion matrix  
#The "No" and "Yes" represent the actual values  
#The "FALSE" and "TRUE" represent our predicted values  
t1 = table(test$violator,predictions > 0.5)  
t1

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

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

## [1] 0.922619

That actually has a very high accuracy rate.