M3\_Assignment2

Corinne Walker

5/27/2021

#install.packages("e1071")  
#install.packages("ROCR")  
  
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.1.1 v dplyr 1.0.5  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(tidymodels)

## -- Attaching packages -------------------------------------- tidymodels 0.1.3 --

## v broom 0.7.6 v rsample 0.1.0   
## v dials 0.0.9 v tune 0.1.5   
## v infer 0.5.4 v workflows 0.2.2   
## v modeldata 0.1.0 v workflowsets 0.0.2   
## v parsnip 0.1.5 v yardstick 0.0.8   
## v recipes 0.1.16

## -- Conflicts ----------------------------------------- tidymodels\_conflicts() --  
## x scales::discard() masks purrr::discard()  
## x dplyr::filter() masks stats::filter()  
## x recipes::fixed() masks stringr::fixed()  
## x dplyr::lag() masks stats::lag()  
## x yardstick::spec() masks readr::spec()  
## x recipes::step() masks stats::step()  
## \* Use tidymodels\_prefer() to resolve common conflicts.

library(e1071)

##   
## Attaching package: 'e1071'

## The following object is masked from 'package:tune':  
##   
## tune

## The following object is masked from 'package:rsample':  
##   
## permutations

library(ROCR)  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

library(glmnet)

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

## Loaded glmnet 4.1-1

library(readr)  
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 %>%   
 rename(gender = male) %>%   
 mutate(gender = as\_factor(gender)) %>%   
 mutate(gender = fct\_recode(gender, "female" = "0", "male" = "1")) %>%  
 mutate(race = as\_factor(race)) %>%   
 mutate(race = fct\_recode(race, "not white" = "2", "white" = "1")) %>%  
 mutate(state = as\_factor(state)) %>%   
 mutate(state = fct\_recode(state, "other state" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4")) %>%  
 mutate(crime = as\_factor(crime)) %>%   
 mutate(crime = fct\_recode(crime, "other crime" = "1", "larceny" = "2", "drug" = "3", "driving" = "4")) %>%  
 mutate(multiple.offenses = as\_factor(multiple.offenses)) %>%   
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "single" = "0", "multiple" = "1")) %>%  
 mutate(violator = as\_factor(violator)) %>%   
 mutate(violator = fct\_recode(violator, "no violation" = "0", "violated" = "1"))

summary(parole)

## gender race age state time.served   
## female:130 white :389 Min. :18.40 other state:143 Min. :0.000   
## male :545 not white: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 single :313 other crime:315 no violation:597   
## 1st Qu.:12.00 multiple:362 larceny :106 violated : 78   
## Median :12.00 drug :153   
## Mean :13.06 driving :101   
## 3rd Qu.:15.00   
## Max. :18.00

## Task 1

Split the data (training and testing). 70% of the data to training. Stratified the random split by the response variable “violator”.

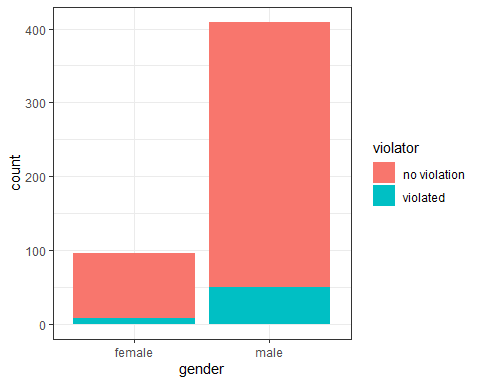
set.seed(12345)  
parole\_split = initial\_split(parole, prob = 0.70, strata = violator)  
train = training(parole\_split)  
test = testing(parole\_split)

## Task 2

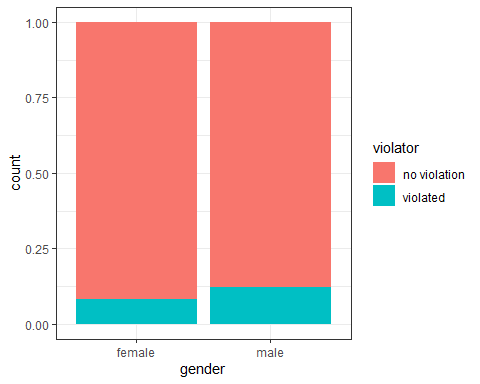
Predict whether or not a parolee will violate his/her parole. ### Visuals gender, race, state, crime, multiple.offenses

**Gender**

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

 Alternative (100% stacked)

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



Tabular data

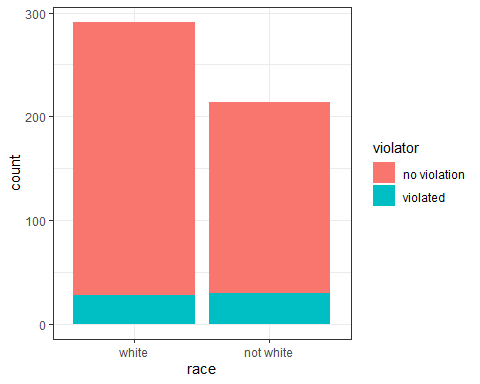
t1 = table(train$violator, train$gender) #create a table object  
prop.table(t1, margin = 2 ) #crosstab with proportions

##   
## female male  
## no violation 0.91666667 0.87775061  
## violated 0.08333333 0.12224939

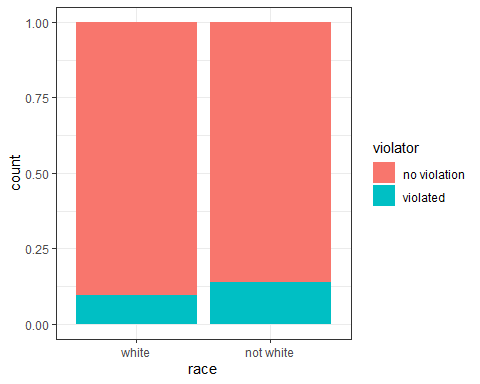
Males have a 3.9% higher likelihood of violating parole when compared evenly to females; however, since the vast majority of people in this dataset are males, there is a higher chance that a parole violator will be male.

**Race**

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



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



Tabular data

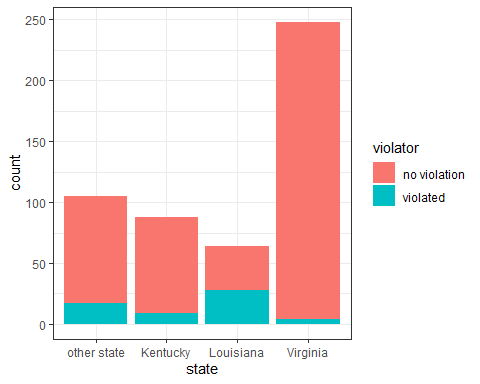
t1 = table(train$violator, train$race)   
prop.table(t1, margin = 2 )

##   
## white not white  
## no violation 0.90378007 0.85981308  
## violated 0.09621993 0.14018692

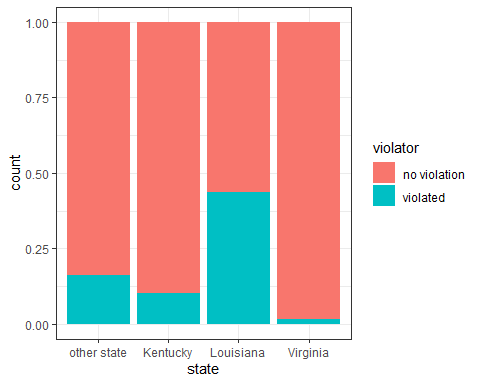
There are more people classified as white in this dataset but “not white” people are slightly more likely to violate parole.

**State**

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



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



Tabular data

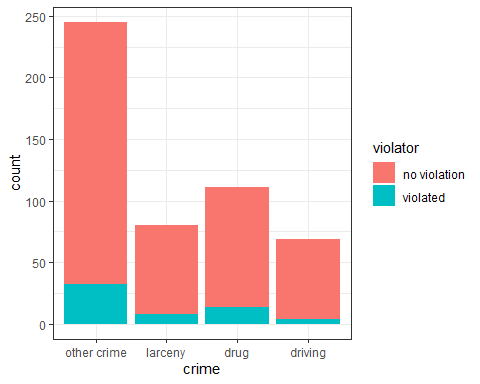
t1 = table(train$violator, train$state)   
prop.table(t1, margin = 2 )

##   
## other state Kentucky Louisiana Virginia  
## no violation 0.83809524 0.89772727 0.56250000 0.98387097  
## violated 0.16190476 0.10227273 0.43750000 0.01612903

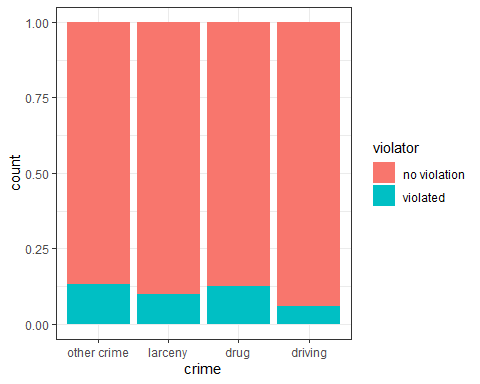
While Virginia has the most people on parole, people in Louisiana have a much higher probability of breaking parole. The state looks like it influenes the “violator” variable.

**Crime**

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



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



Tabular data

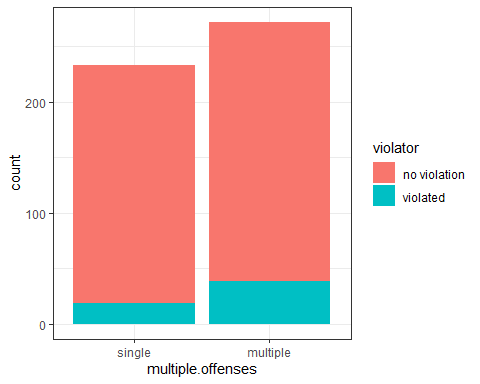
t1 = table(train$violator, train$crime)   
prop.table(t1, margin = 2 )

##   
## other crime larceny drug driving  
## no violation 0.86938776 0.90000000 0.87387387 0.94202899  
## violated 0.13061224 0.10000000 0.12612613 0.05797101

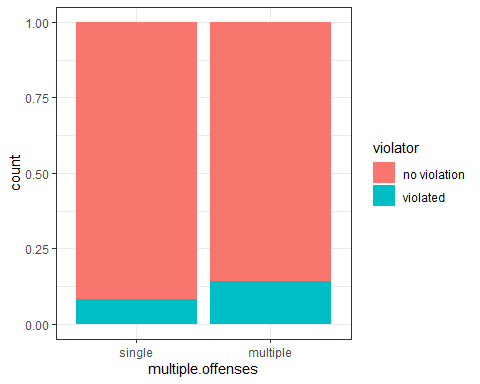
“Other crime” is more likely to have a parole violation because there is a higher number of people that fall in that category. The type of crime does not appear to have a large influence on “violator”.

**Multiple Offenses**

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



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



Tabular data

t1 = table(train$violator, train$multiple.offenses)   
prop.table(t1, margin = 2 )

##   
## single multiple  
## no violation 0.91845494 0.85661765  
## violated 0.08154506 0.14338235

People with multiple offenses have a higher likelihood of violating parole.

## Task 3

The variables “race”, “state”, and “multiple.offenses” seem to be important predictors of “violator”. When comparing all of the variables, “state” appears to have the largest influence on “violator” as a predictor of whether or not someone will break parole.

Logistic regression model with state

parole\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
parole\_recipe = recipe(violator ~ state, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe) %>%   
 add\_model(parole\_model)  
  
parole\_fit1 = fit(logreg\_wf, parole)

summary(parole\_fit1$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.0955 -0.4981 -0.2071 -0.2071 2.7760   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.8165 0.2411 -7.534 4.92e-14 \*\*\*  
## state\_Kentucky -0.2079 0.3728 -0.558 0.577   
## state\_Louisiana 1.6207 0.3277 4.946 7.58e-07 \*\*\*  
## state\_Virginia -2.0153 0.4517 -4.461 8.15e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 483.27 on 674 degrees of freedom  
## Residual deviance: 382.89 on 671 degrees of freedom  
## AIC: 390.89  
##   
## Number of Fisher Scoring iterations: 6

The model shows that the states Louisiana and Virginia, along with “other state”, are very significant to predicting “violator”. Louisiana has a positive coefficient meaning that there is a higher probability of violating parole if the person in from this state. The AIC of this model (a measure of model quality) is 390.89.

Logistic regression model with gender, race, state, crime, multiple.offenses

parole\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
parole\_recipe = recipe(violator ~ gender + race + state + crime + multiple.offenses, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe) %>%   
 add\_model(parole\_model)  
  
parole\_fit2 = fit(logreg\_wf, parole)

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

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.5968 -0.3974 -0.2598 -0.1570 2.9083   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.74215 0.47264 -5.802 6.56e-09 \*\*\*  
## gender\_male 0.32572 0.36762 0.886 0.3756   
## race\_not.white 0.76012 0.32104 2.368 0.0179 \*   
## state\_Kentucky 0.13671 0.41244 0.331 0.7403   
## state\_Louisiana 0.73326 0.39741 1.845 0.0650 .   
## state\_Virginia -3.22589 0.52179 -6.182 6.32e-10 \*\*\*  
## crime\_larceny 0.36011 0.42223 0.853 0.3937   
## crime\_drug -0.25802 0.35458 -0.728 0.4668   
## crime\_driving -0.08241 0.47660 -0.173 0.8627   
## multiple.offenses\_multiple 1.51015 0.32166 4.695 2.67e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 483.27 on 674 degrees of freedom  
## Residual deviance: 351.00 on 665 degrees of freedom  
## AIC: 371  
##   
## Number of Fisher Scoring iterations: 6

The variables gender, race, state, crime, and multiple.offenses were added to the model so see how they would influence it. While the AIC did improve, 371 compared to 390.89 from the previous model, not all of the variables used were statistically significant. Being female and from Virginia were significant to predicting that the person would no break parole. Having a race of “not white” and multiple offenses had a significant and very significant probability of predicting that someone would break parole, respectively. Interestingly, when all of the variables were compared, Louisiana became insignificant as a predictor.

Logistic regression model with gender, race, state, multiple.offenses

parole\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
parole\_recipe = recipe(violator ~ gender + race + state + multiple.offenses, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe) %>%   
 add\_model(parole\_model)  
  
parole\_fit3 = fit(logreg\_wf, parole)

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

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.4252 -0.4121 -0.2655 -0.1844 2.8576   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.72593 0.43253 -6.302 2.93e-10 \*\*\*  
## gender\_male 0.25945 0.35599 0.729 0.4661   
## race\_not.white 0.73836 0.31819 2.320 0.0203 \*   
## state\_Kentucky 0.08509 0.39807 0.214 0.8307   
## state\_Louisiana 0.77045 0.39291 1.961 0.0499 \*   
## state\_Virginia -3.12308 0.51096 -6.112 9.83e-10 \*\*\*  
## multiple.offenses\_multiple 1.52366 0.32132 4.742 2.12e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 483.27 on 674 degrees of freedom  
## Residual deviance: 352.71 on 668 degrees of freedom  
## AIC: 366.71  
##   
## Number of Fisher Scoring iterations: 6

The variable “crime” was removed from the model since it showed no significance. This improved the AIC from 371 to 366.71. Being female and from Virginia are significant predictors that someone will not break parole. Being “not white” and from Louisiana are significant predictors along with having multiple offenses, which is a very significant predictor, that someone will break parole. The model seems fairly intuitive based off of the information gleaned from the above graphs that were used to visualize the variables.

## Task 5

Logistic regression model with race, state, multiple.offenses

parole\_model =   
 logistic\_reg() %>% #note the use of logistic\_reg  
 set\_engine("glm") #standard logistic regression engine is glm  
  
parole\_recipe = recipe(violator ~ race + state + multiple.offenses, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) #exclude the response variable from being dummy converted   
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe) %>%   
 add\_model(parole\_model)  
  
parole\_fit4 = fit(logreg\_wf, parole)

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

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.4012 -0.4051 -0.2604 -0.1801 2.8739   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.50359 0.30055 -8.330 < 2e-16 \*\*\*  
## race\_not.white 0.74594 0.31828 2.344 0.0191 \*   
## state\_Kentucky 0.04449 0.39449 0.113 0.9102   
## state\_Louisiana 0.75016 0.39147 1.916 0.0553 .   
## state\_Virginia -3.12945 0.51147 -6.119 9.44e-10 \*\*\*  
## multiple.offenses\_multiple 1.51964 0.32027 4.745 2.09e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 483.27 on 674 degrees of freedom  
## Residual deviance: 353.26 on 669 degrees of freedom  
## AIC: 365.26  
##   
## Number of Fisher Scoring iterations: 6

THe AIC for this model 365.26 which is minutely better than the AIC of 366.71 from the model that also included gender. Being white and from Virginia are very significant predictors that someone would not break parole. Being “not white” and having multiple offenses are significant and very significant predictors that someone would break parole, respectively. Louisiana became insignificant.

## Task 6

Prediction: Parolee1- Louisiana with multiple offenses and white race

newdata = data.frame(state = "Louisiana", multiple.offenses = "multiple", race = "white")  
predict(parole\_fit4, newdata, type="prob")

## # A tibble: 1 x 2  
## `.pred\_no violation` .pred\_violated  
## <dbl> <dbl>  
## 1 0.558 0.442

A parolee that is white with multiple offenses and from Louisiana has a 55.8% chance of not breaking parole and a 44.2% chance of violating parole.

Prediction: Parolee1- Kentucky with no multiple offenses and other race

newdata = data.frame(state = "Kentucky", multiple.offenses = "single", race = "not white")  
predict(parole\_fit4, newdata, type="prob")

## # A tibble: 1 x 2  
## `.pred\_no violation` .pred\_violated  
## <dbl> <dbl>  
## 1 0.847 0.153

A parolee that is not white with no multiple offenses and from Kentucky has a 84.7% chance of not breaking parole and a 15.3% chance of violating parole.

## Task 7

ROC curve and probability threshold that best balances specificity and sensitivity on training set

Develop predictions

predictions = predict(parole\_fit4, train, type="prob") #develop predicted probabilities  
head(predictions)

## # A tibble: 6 x 2  
## `.pred\_no violation` .pred\_violated  
## <dbl> <dbl>  
## 1 0.924 0.0756  
## 2 0.853 0.147   
## 3 0.924 0.0756  
## 4 0.853 0.147   
## 5 0.924 0.0756  
## 6 0.924 0.0756

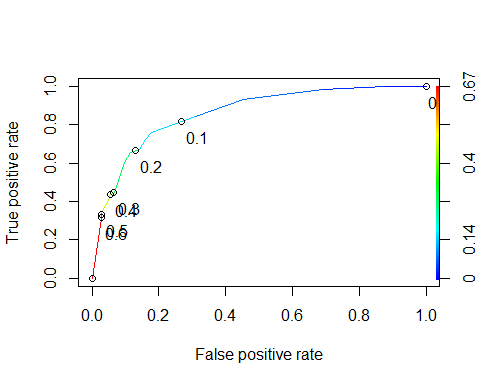
Extract just the “Yes” prediction.

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

## # A tibble: 6 x 1  
## .pred\_violated  
## <dbl>  
## 1 0.0756  
## 2 0.147   
## 3 0.0756  
## 4 0.147   
## 5 0.0756  
## 6 0.0756

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))



Area under the curve (AUC). AUC is a measure of the strength of the model. Values closer to 1 are better. Can be used to compare models.

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

## [1] 0.8590218

## 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 0.7586207  
## specificity 0.8210291  
## cutoff 0.1470858

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 > 0.1470858)  
t1

##   
## FALSE TRUE  
## no violation 376 71  
## violated 16 42

Calculate accuracy

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

## [1] 0.8277228

The implications that could occur with incorrectly classifying a parolee as low risk when they are actually high rist is that they might not receive the proper attention and support to keep them from violating their parole. Additionally, if someone is labeled as high risk when they actually low risk, unnecessary resources might be spent or allotted to them that would have been better used on a high risk parolee.

## Task 9

Identify a probability threshold (via trial-and-error) that best maximizes accuracy on the training set

Apply trial and error to maximize accuracy (here trying 0.5 as threshold)

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

##   
## FALSE TRUE  
## no violation 435 12  
## violated 40 18

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

## [1] 0.8970297

Threshold = 0.6

t2 = table(train$violator,predictions > 0.6)  
t2

##   
## FALSE TRUE  
## no violation 435 12  
## violated 40 18

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

## [1] 0.8970297

I received a “subscript out of bounds” error when I tried to run 0.7 for a threshold so I’m choosing 0.6 as the threshold that best maximizes accuracy on the training set.

## Task 10

Calculate accuracy with the probability threshold from Task 9

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

## [1] 0.8970297

The model has a 89.7% accuracy on the testing set.