library(tidyverse)

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

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.6 v dplyr 1.0.7  
## v tidyr 1.1.4 v stringr 1.4.0  
## v readr 2.1.1 v forcats 0.5.1

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

library(tidymodels)

## Registered S3 method overwritten by 'tune':  
## method from   
## required\_pkgs.model\_spec parsnip

## -- Attaching packages -------------------------------------- tidymodels 0.1.4 --

## v broom 0.7.11 v rsample 0.1.1   
## v dials 0.0.10 v tune 0.1.6   
## v infer 1.0.0 v workflows 0.2.4   
## v modeldata 0.1.1 v workflowsets 0.1.0   
## v parsnip 0.1.7 v yardstick 0.0.9   
## v recipes 0.1.17

## -- 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()  
## \* Dig deeper into tidy modeling with R at https://www.tmwr.org

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)

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

## Rows: 675 Columns: 9

## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## dbl (9): male, race, age, state, time.served, max.sentence, multiple.offense...

##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

str(parole)

## spec\_tbl\_df [675 x 9] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ male : num [1:675] 1 0 1 1 1 1 1 0 0 1 ...  
## $ race : num [1:675] 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num [1:675] 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ state : num [1:675] 1 1 1 1 1 1 1 1 1 1 ...  
## $ time.served : num [1:675] 5.5 5.4 5.6 5.7 5.4 6 6 4.8 4.5 4.7 ...  
## $ max.sentence : num [1:675] 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: num [1:675] 0 0 0 0 0 0 0 0 0 0 ...  
## $ crime : num [1:675] 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : num [1:675] 0 0 0 0 0 0 0 0 0 0 ...  
## - attr(\*, "spec")=  
## .. 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()  
## .. )  
## - attr(\*, "problems")=<externalptr>

summary(parole)

## male race age state   
## Min. :0.0000 Min. :1.000 Min. :18.40 Min. :1.000   
## 1st Qu.:1.0000 1st Qu.:1.000 1st Qu.:25.35 1st Qu.:2.000   
## Median :1.0000 Median :1.000 Median :33.70 Median :3.000   
## Mean :0.8074 Mean :1.424 Mean :34.51 Mean :2.887   
## 3rd Qu.:1.0000 3rd Qu.:2.000 3rd Qu.:42.55 3rd Qu.:4.000   
## Max. :1.0000 Max. :2.000 Max. :67.00 Max. :4.000   
## time.served max.sentence multiple.offenses crime   
## Min. :0.000 Min. : 1.00 Min. :0.0000 Min. :1.000   
## 1st Qu.:3.250 1st Qu.:12.00 1st Qu.:0.0000 1st Qu.:1.000   
## Median :4.400 Median :12.00 Median :1.0000 Median :2.000   
## Mean :4.198 Mean :13.06 Mean :0.5363 Mean :2.059   
## 3rd Qu.:5.200 3rd Qu.:15.00 3rd Qu.:1.0000 3rd Qu.:3.000   
## Max. :6.000 Max. :18.00 Max. :1.0000 Max. :4.000   
## violator   
## Min. :0.0000   
## 1st Qu.:0.0000   
## Median :0.0000   
## Mean :0.1156   
## 3rd Qu.:0.0000   
## Max. :1.0000

parole = parole %>% mutate(male = as\_factor(male)) %>%   
 mutate(male = fct\_recode(male, "female" = "0", "male" = "1" ))   
parole = parole %>% mutate(race = as\_factor(race)) %>%   
 mutate(race = fct\_recode(race, "non-white" = "2", "white" = "1" ))   
parole = parole %>% mutate(state = as\_factor(state)) %>%   
 mutate(state = fct\_recode(state, "other" = "1", "Kentucky" = "2", "Louisiana" ="3", "Virginia" = "4" ))   
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(multiple.offenses = as\_factor(multiple.offenses)) %>%   
 mutate(male = fct\_recode(multiple.offenses, "no" = "0", "yes" = "1" ))  
parole = parole %>% mutate(violator = as\_factor(violator)) %>%   
 mutate(male = fct\_recode(violator, "no" = "0", "yes" = "1" ))  
str(parole)

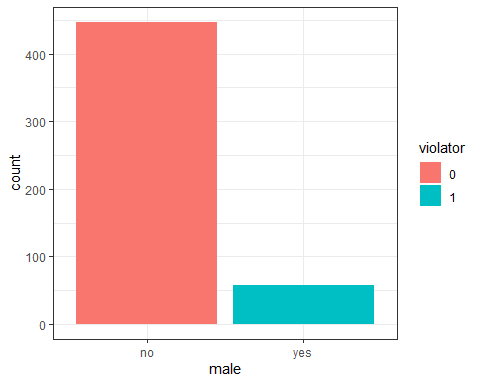
## spec\_tbl\_df [675 x 9] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ male : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ race : Factor w/ 2 levels "white","non-white": 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num [1:675] 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ state : Factor w/ 4 levels "other","Kentucky",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ time.served : num [1:675] 5.5 5.4 5.6 5.7 5.4 6 6 4.8 4.5 4.7 ...  
## $ max.sentence : num [1:675] 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "other","larceny",..: 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## - attr(\*, "spec")=  
## .. 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()  
## .. )  
## - attr(\*, "problems")=<externalptr>

Split!

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

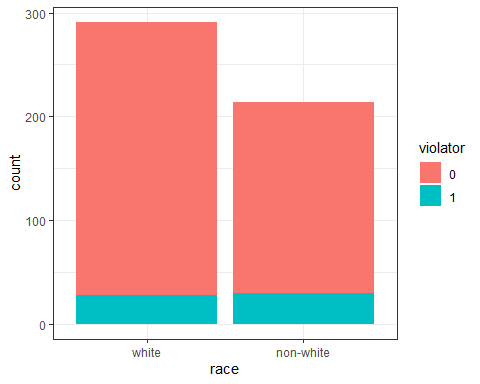
Some visualizations to help illustrate which variables affect violating parole.

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



It appears women tend to violate their parole more than men do.

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

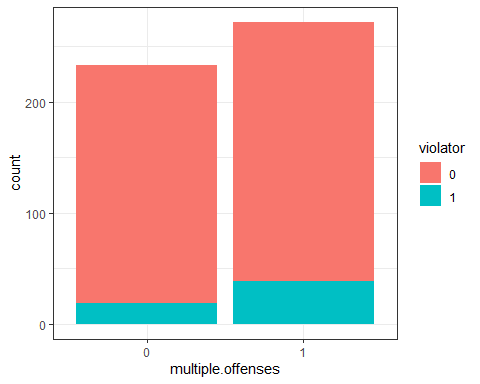


Race does not appear to play a massive role, but non-whites as a percentage would violate their parole more. 14% non-white to 9% white. +5% rough increase based on race.

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

##   
## white non-white  
## 0 0.90488432 0.85664336  
## 1 0.09511568 0.14335664

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

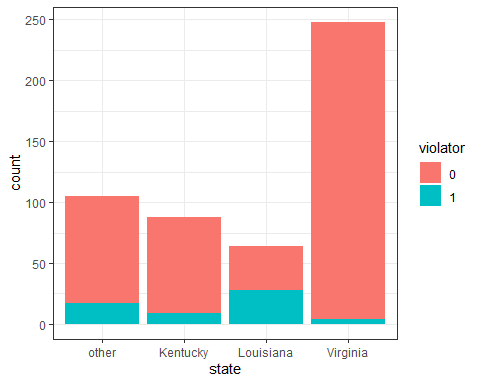


t2 = table(parole$violator,parole$multiple.offenses)  
prop.table(t2, margin = 2)

##   
## 0 1  
## 0 0.9201278 0.8535912  
## 1 0.0798722 0.1464088

Similar to race, you see a slight increase (about 5% roughly) in relation to violating parole. If you have been incarcerated for multiple offenses, you ARE considered slightly more likely to violate your parole.

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



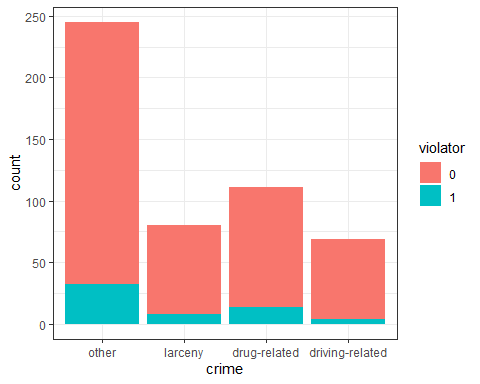
In a surprise, state seems to have a larger impact than you might expect (without reading the rationale). Louisana appears to be a large contributor, geographically, of determining/predicting if parole will be violated. Let’s confirm in a graph.

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

##   
## other Kentucky Louisiana Virginia  
## 0 0.86013986 0.88333333 0.54878049 0.97878788  
## 1 0.13986014 0.11666667 0.45121951 0.02121212

Slightly over 45% of the violators come from Louisana, whereas you are less likely to violate parole based on this variable if you are from Virginia, for example (or Kentucky/other)!

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



The type of crime does not appear to have a massive showcasing for indicating what will premeditate parole violations. The three outlined (larceny, drugs, driving) are all pretty similar.

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

##   
## other larceny drug-related driving-related  
## 0 0.87619048 0.87735849 0.87581699 0.93069307  
## 1 0.12380952 0.12264151 0.12418301 0.06930693

Based on all of these variables, I would choose the state variable to best help evaluate and predict our response variable, Violator. There appears to be a stronger relationship (than the other variables here) with the state you reside from in determining parole violations. This could make sense due to various socio-economic reasons, but also laws, legal loopholes, and parole standards differing state to state (for one idea). Let’s create a logistic regression model to illustrate this point.

parole\_model =   
 logistic\_reg(mode = "classification") %>% #note the use of logistic\_reg and mode = "classification"  
 set\_engine("glm") #standard logistic regression engine is 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   
## -1.0727 -0.4645 -0.1803 -0.1803 2.8730   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.6441 0.2649 -6.206 5.44e-10 \*\*\*  
## stateKentucky -0.5281 0.4404 -1.199 0.230480   
## stateLouisiana 1.3928 0.3656 3.809 0.000139 \*\*\*  
## stateVirginia -2.4668 0.5693 -4.333 1.47e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 360.11 on 504 degrees of freedom  
## Residual deviance: 279.75 on 501 degrees of freedom  
## AIC: 287.75  
##   
## Number of Fisher Scoring iterations: 6

The AIC of this model is 287.75, which isn’t extraordinarily high but perhaps could be lowered. Virginia and Louisana appear to be signiificant here. Louisiana indicating that you would be more likely to violate parole if you are from there versus Virginia indicating that you would be far less likely. Kentucky does not appear to be statistically significant here, but the est. std has a negative correlation to commiting parole violations.

Let’s try to find the most optimal model with AIC as our evaluator.

parole\_model2 =   
 logistic\_reg(mode = "classification") %>% #note the use of logistic\_reg and mode = "classification"  
 set\_engine("glm") #standard logistic regression engine is glm  
  
parole\_recipe2 = recipe(violator ~ crime, train)  
  
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   
## -0.5291 -0.5291 -0.5193 -0.4590 2.3866   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.8956 0.1896 -9.998 <2e-16 \*\*\*  
## crimelarceny -0.3017 0.4181 -0.721 0.471   
## crimedrug-related -0.0401 0.3431 -0.117 0.907   
## crimedriving-related -0.8925 0.5489 -1.626 0.104   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 360.11 on 504 degrees of freedom  
## Residual deviance: 356.59 on 501 degrees of freedom  
## AIC: 364.59  
##   
## Number of Fisher Scoring iterations: 5

AIC increases here in this case.

parole\_model3 =   
 logistic\_reg(mode = "classification") %>% #note the use of logistic\_reg and mode = "classification"  
 set\_engine("glm") #standard logistic regression engine is glm  
  
parole\_recipe3 = recipe(violator ~ race + crime, train)  
  
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   
## -0.5860 -0.4990 -0.4807 -0.4089 2.4482   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.099845 0.241493 -8.695 <2e-16 \*\*\*  
## racenon-white 0.419116 0.283259 1.480 0.139   
## crimelarceny -0.339958 0.419810 -0.810 0.418   
## crimedrug-related 0.005668 0.345318 0.016 0.987   
## crimedriving-related -0.845749 0.550491 -1.536 0.124   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 360.11 on 504 degrees of freedom  
## Residual deviance: 354.40 on 500 degrees of freedom  
## AIC: 364.4  
##   
## Number of Fisher Scoring iterations: 5

Race and crime together also are about the same as crime alone. What if we take all of the variables?

parole\_model4 =   
 logistic\_reg(mode = "classification") %>% #note the use of logistic\_reg and mode = "classification"  
 set\_engine("glm") #standard logistic regression engine is glm  
  
parole\_recipe4 = recipe(violator ~., train)  
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe4) %>%   
 add\_model(parole\_model4)  
  
parole\_fit4 = fit(logreg\_wf, train)

## Warning: glm.fit: algorithm did not converge

options(scipen = 999) #optional, but suppresses scientific notation  
summary(parole\_fit4$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.000002409 -0.000002409 -0.000002409 -0.000002409 0.000002409   
##   
## Coefficients:  
## Estimate Std. Error  
## (Intercept) -26.566068523537101242 115560.113860363140702248  
## maleyes 53.132137547552993340 56648.028350091743050143  
## racenon-white -0.000000000000173640 36011.996786441508447751  
## age -0.000000000000007516 1584.126748150792082015  
## stateKentucky -0.000000000000761446 54836.449668094399385154  
## stateLouisiana -0.000000000003611966 69044.086725114801083691  
## stateVirginia -0.000000000000783227 52745.245120707135356497  
## time.served 0.000000000000209676 13388.758934958117606584  
## max.sentence -0.000000000000060201 6067.432244601792262984  
## multiple.offenses1 0.000000000000127467 39954.110743762452329975  
## crimelarceny -0.000000000000076744 46423.752574692334746942  
## crimedrug-related -0.000000000000198305 43273.756296548031968996  
## crimedriving-related 0.000000000000107385 49919.073397755171754397  
## z value Pr(>|z|)  
## (Intercept) 0.000 1.000  
## maleyes 0.001 0.999  
## racenon-white 0.000 1.000  
## age 0.000 1.000  
## stateKentucky 0.000 1.000  
## stateLouisiana 0.000 1.000  
## stateVirginia 0.000 1.000  
## time.served 0.000 1.000  
## max.sentence 0.000 1.000  
## multiple.offenses1 0.000 1.000  
## crimelarceny 0.000 1.000  
## crimedrug-related 0.000 1.000  
## crimedriving-related 0.000 1.000  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 360.1052407601982 on 504 degrees of freedom  
## Residual deviance: 0.0000000029298 on 492 degrees of freedom  
## AIC: 26  
##   
## Number of Fisher Scoring iterations: 25

AIC of 26 is quite impressive. I do not anticipate finding a combination with an AIC lower than 26. Looking at this model, we can see being male can have a positive correlation on committing parole violations. However, this models estimate std.’s are all actually all extremely close together and not particularly useful. Grouping all of the variables to predict violators makes this model hard to read and comprehend. Based on the task, this appears to be the lowest AIC, but that may not mean that this is really all that clean and intuitive. Perhaps limiting the run to 3-4 variables would bring a higher AIC but much more clarity that is needed to continue effective analysis.

Let’s try a new regression model with some select variables.

parole\_model5 =   
 logistic\_reg(mode = "classification") %>% #note the use of logistic\_reg and mode = "classification"  
 set\_engine("glm") #standard logistic regression engine is glm  
  
parole\_recipe5 = recipe(violator ~ state + multiple.offenses + race, train)  
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe5) %>%   
 add\_model(parole\_model5)  
  
parole\_fit5 = fit(logreg\_wf, train)  
  
summary(parole\_fit5$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.3821 -0.4502 -0.2291 -0.1494 3.0002   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.2382 0.3202 -6.990 0.00000000000275 \*\*\*  
## stateKentucky -0.4641 0.4627 -1.003 0.3159   
## stateLouisiana 0.3328 0.4577 0.727 0.4671   
## stateVirginia -3.7640 0.6526 -5.768 0.00000000802616 \*\*\*  
## multiple.offenses1 1.5128 0.3704 4.084 0.00004425741578 \*\*\*  
## racenon-white 0.8620 0.3766 2.288 0.0221 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 360.11 on 504 degrees of freedom  
## Residual deviance: 257.04 on 499 degrees of freedom  
## AIC: 269.04  
##   
## Number of Fisher Scoring iterations: 7

There is an AIC of 269, which is better than some of our prior attempts, but not as good as 26. However, this is much clearer than our prior model. It does not look like Kentucky, or Louisiana are actually significant here, despite my early indications! It appears having multiple offenses and being a non-white offender seem to increase the prediction/ your assumed chance of being a parole violator while being from Virginia appears to have a negative correlation, lowering the predictions chance’s of classifying you as a parole violator. Overall this model is still sub 300 on AIC and provides some more clear insight into where we need to focus our efforts on finding the best affecting variables on “violators”.

Let’s make some predictions based off of our variables; some sample scenarios.

newdata = data.frame(state = "Louisiana", multiple.offenses = "1", race = "white")  
predict(parole\_fit5, newdata, type = "prob")

## # A tibble: 1 x 2  
## .pred\_0 .pred\_1  
## <dbl> <dbl>  
## 1 0.597 0.403

With this combination, it appears that we predict you are more likely to NOT violate your parole based on the combination of these variables. Let’s try another. (Bordering on 60/40 no/yes split here)

newdata2 = data.frame(state = "Kentucky", multiple.offenses = "0", race = "non-white")  
predict(parole\_fit5, newdata2, type = "prob")

## # A tibble: 1 x 2  
## .pred\_0 .pred\_1  
## <dbl> <dbl>  
## 1 0.863 0.137

With this combination, you are predicted to be even less likely to violate your parole (86% roughly no/yes)!

Now let’s create a ROC curve to better illustrate some points and illuiminate/discover our probability threshold that best balances specifcity alongside sensitivity.

parole\_model5 =   
 logistic\_reg(mode = "classification") %>% #note the use of logistic\_reg and mode = "classification"  
 set\_engine("glm") #standard logistic regression engine is glm  
  
parole\_recipe5 = recipe(violator ~ state + multiple.offenses + race, train)  
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe5) %>%   
 add\_model(parole\_model5)  
  
parole\_fit5 = fit(logreg\_wf, train)

( Just to remind to utilize the model with all of the variables)

predictions = predict(parole\_fit5, train, type = "prob")  
head(predictions)

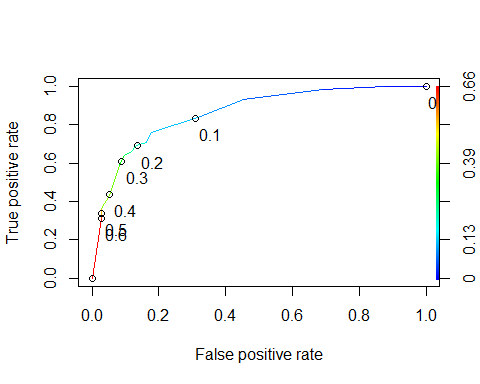
## # A tibble: 6 x 2  
## .pred\_0 .pred\_1  
## <dbl> <dbl>  
## 1 0.904 0.0964  
## 2 0.798 0.202   
## 3 0.904 0.0964  
## 4 0.798 0.202   
## 5 0.904 0.0964  
## 6 0.904 0.0964

predictions = predict(parole\_fit5, train, type="prob")[2] #Pulls column #2, the YES  
head(predictions)

## # A tibble: 6 x 1  
## .pred\_1  
## <dbl>  
## 1 0.0964  
## 2 0.202   
## 3 0.0964  
## 4 0.202   
## 5 0.0964  
## 6 0.0964

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] 0.8606804

#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 0.1295001

According to our model, the cut off is just shy of 13% at 12.9%, which best balances specificity and sensitivity. Let’s now calculate the accuracy, specificity, and sensitivity based off the model on the training set!

#confusion matrix  
#The "No" and "Yes" represent the actual values  
#The "FALSE" and "TRUE" represent our predicted values  
t1 = table(train$violator,predictions > 0.1295001)  
t1

##   
## FALSE TRUE  
## 0 374 73  
## 1 17 41

Accuracy:

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

## [1] 0.8217822

Using the 12.9% cutoff, we predict we will be 82.1% accurate.

Sensitivity

41/(17+41)

## [1] 0.7068966

The sensitivity of the model is 70.6%.

Specificity

374/(73+374)

## [1] 0.836689

The specificity of the model is 83.6%. If we were to incorrectly classify a parolee, it could ruin their lives first and foremost. If we only take accuracy into account, we could conversely miss a parolee who did violate their parole! Incorrectly classifying or being negligent to the point of enabling false negatives to go off free of consequence would have major social implications alongside the legality of it all. It’s always a good idea to cover your bases, especially in data analysis. It makes sense to take precaution and care so we do not misclassify a parolee. If they violate their parole and go unpunished, the system is now inherently effete and broken. If we misclassify a parolee who did NOT violate their parole as an offender, it could ruin their lives.

Let’s continue and find a probability threshold that best maximizes accuracy on this set.

t2 = table(train$violator,predictions > 0.50)  
t2

##   
## FALSE TRUE  
## 0 435 12  
## 1 39 19

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

## [1] 0.8990099

At a .5 threshold, we’re looking at almost 90% accurate. Threshold = 0.6

t3 = table(train$violator,predictions > 0.6)  
t3

##   
## FALSE TRUE  
## 0 435 12  
## 1 40 18

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

## [1] 0.8970297

Going up to .6 leads us to a slightly worse accuracy of 89.7 versus the prior 89.9%.

t4 = table(train$violator,predictions > 1) #set threshold to 1 so all are classified as not delinquent  
t4

##   
## FALSE  
## 0 447  
## 1 58

(t4[1])/nrow(train)

## [1] 0.8851485

The >1 threshold gives us an even lower threshold of 88.5% here. Accuracy won’t always tell the entire story, sensitivity and specificity will help us decipher and analyze our thresholds/models more effectively.

If we use the <.5 threshold on our testing set, here are the results.

parole\_model6 =   
 logistic\_reg(mode = "classification") %>% #note the use of logistic\_reg and mode = "classification"  
 set\_engine("glm") #standard logistic regression engine is glm  
  
parole\_recipe6 = recipe(violator ~ state + multiple.offenses + race, test)  
  
logreg\_wf = workflow() %>%  
 add\_recipe(parole\_recipe6) %>%   
 add\_model(parole\_model6)  
  
parole\_fit6 = fit(logreg\_wf, test)

predictions2 = predict(parole\_fit6, test, type="prob")[2] #Pulls column #2, the YES  
head(predictions2)

## # A tibble: 6 x 1  
## .pred\_1  
## <dbl>  
## 1 0.0215  
## 2 0.0398  
## 3 0.0215  
## 4 0.257   
## 5 0.0398  
## 6 0.0215

t4 = table(test$violator,predictions2 > 0.5)  
t4

##   
## FALSE TRUE  
## 0 145 5  
## 1 12 8

(t4[1,1]+t4[2,2])/nrow(test)

## [1] 0.9

Based on the .5 probability threshold here, we get an accuracy reporting of 90%. This is almost identical to our training set.