# Jon Bailey

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ ggplot2 3.3.2 ✓ purrr 0.3.4  
## ✓ tibble 3.0.4 ✓ dplyr 1.0.2  
## ✓ tidyr 1.1.2 ✓ stringr 1.4.0  
## ✓ readr 1.4.0 ✓ forcats 0.5.0

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

library(tidymodels)

## ── Attaching packages ────────────────────────────────────── tidymodels 0.1.2 ──

## ✓ broom 0.7.4 ✓ recipes 0.1.15  
## ✓ dials 0.0.9 ✓ rsample 0.0.8   
## ✓ infer 0.5.4 ✓ tune 0.1.2   
## ✓ modeldata 0.1.0 ✓ workflows 0.2.1   
## ✓ parsnip 0.1.5 ✓ yardstick 0.0.7

## ── 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()

library(e1071)

##   
## Attaching package: 'e1071'

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

library(ROCR)  
library(glmnet)

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

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

## Loaded glmnet 4.1

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()  
## )

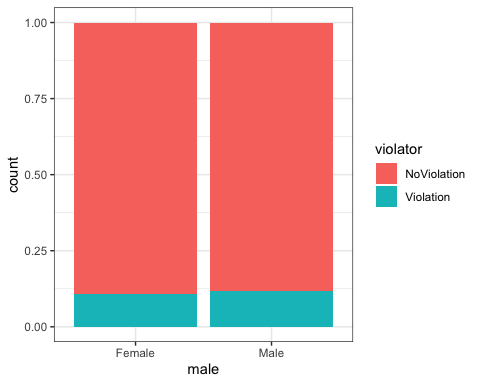
parole = parole %>%  
 mutate(male = as\_factor(male)) %>%  
 mutate(male = fct\_recode(male, "Female" = "0", "Male" = "1")) %>%  
 mutate(race = as\_factor(race)) %>%  
 mutate(race = fct\_recode(race, "White" = "1", "Otherwise" = "2")) %>%  
 mutate(state = as\_factor(state)) %>%  
 mutate(state = fct\_recode(state, "Other" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4")) %>%  
 mutate(crime = as\_factor(crime)) %>%  
 mutate(crime = fct\_recode(crime, "Other" = "1", "Larceny" = "2", "Drugs" = "3", "Driving" = "4")) %>%  
 mutate(multiple.offenses = as\_factor(multiple.offenses)) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "MultipleOffenses" = "1", "Other" = "0")) %>%  
 mutate(violator = as\_factor(violator)) %>%  
 mutate(violator = fct\_recode(violator, "Violation" = "1", "NoViolation" = "0"))

### The above block renamed the variables as factors, and then recoded the categorical variables from numerics into meaningful strings of data.

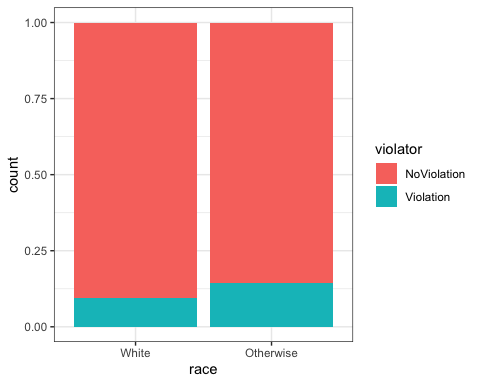
set.seed(12345)  
parole\_split = initial\_split(parole, prob = 0.70, strata = violator)  
train = training(parole\_split)  
test = testing(parole\_split)

### In the test dataset, there are 168 rows and 9 columns. In the train dataset, there are 507 rows and 9 columns.

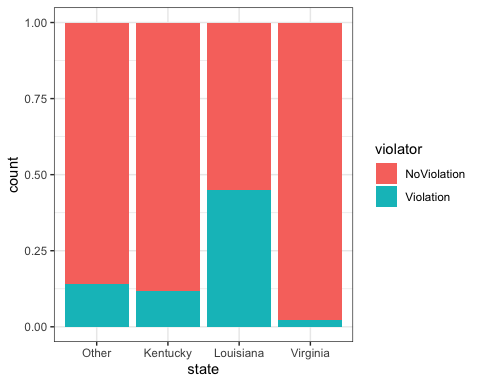
ggplot(parole, aes(x=male, fill = violator)) +  
 geom\_bar(position = "fill") +  
 theme\_bw()



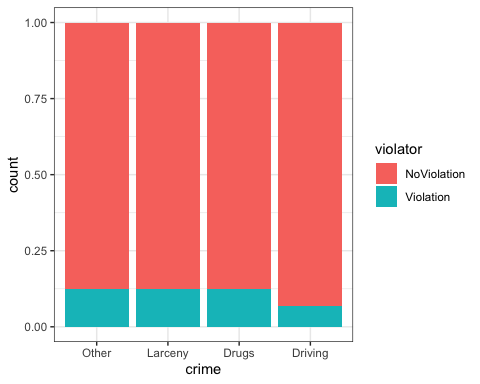
ggplot(parole, aes(x=race, fill = violator)) +  
 geom\_bar(position = "fill") +  
 theme\_bw()



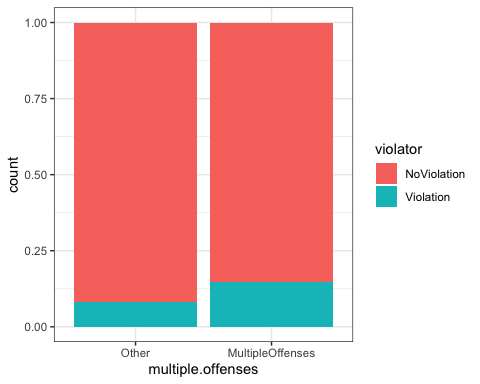
ggplot(parole, aes(x=state, fill = violator)) +  
 geom\_bar(position = "fill") +  
 theme\_bw()



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



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



### The above graphs show slight variations in the data and how these factors relate to whether individuals were likely to break their parole or not. In terms of male or female, these data showed that gender is not a good predictor of whether someone is more likely to break their parole.

### In terms of race, races other than white are predicted to be slightly higher in breaking their parole.

### In terms of state, this is by far the highest predictor of breaking parole, with Louisiana being much higher in being predicted to break parole, and Virginia being much lower in predicted of breaking parole.

### In terms of crime, driving-related crimes are predicted to be the lowest in breaking parole.

### Finally, individuals with multiple offenses are predicted to be slightly higher in breaking their parole.

### The best predictor of the variable violator, or whether or not an individual can be predicted to break their parole, is the variable state.

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

summary(parole\_fit$fit$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

### Note the AIC of this model, which is 390.89, and this is a measure of the model quality. This value is used to measure the quality compared to other models, and a smaller AIC is usually seen as better. In looking at the p-values, both Louisiana and Virginia are very significant, but in different ways. Louisiana is significant in predicting that more individuals from that state are more likely to break their parole, and Virginia is significant in predicting more individuals from that state are less likely to break their parole.

train\_model =   
 logistic\_reg() %>%  
 set\_engine("glm")  
  
train\_recipe = recipe(violator ~ state, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
trainlogreg\_wf = workflow() %>%  
 add\_recipe(train\_recipe) %>%  
 add\_model(train\_model)  
  
train\_fit = fit(trainlogreg\_wf, train)

summary(train\_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

### The AIC of this model, 308.7, is better than the AIC of the logistical regression model of the parole dataset, which means that this model is better quality for predicting.

crime\_model =   
 logistic\_reg() %>%  
 set\_engine("glm")  
  
crime\_recipe = recipe(violator ~ crime, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
crimelogreg\_wf = workflow() %>%  
 add\_recipe(crime\_recipe) %>%  
 add\_model(crime\_model)  
  
crime\_fit = fit(crimelogreg\_wf, train)

summary(crime\_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\_Drugs -0.030054 0.347904 -0.086 0.931   
## crime\_Driving -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

alltrain\_model =   
 logistic\_reg() %>%  
 set\_engine("glm")  
  
alltrain\_recipe = recipe(violator ~ male + race + state + crime + multiple.offenses, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
alltrainlogreg\_wf = workflow() %>%  
 add\_recipe(alltrain\_recipe) %>%  
 add\_model(alltrain\_model)  
  
alltrain\_fit = fit(alltrainlogreg\_wf, train)

summary(alltrain\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.4698 -0.4378 -0.2718 -0.1801 2.8228   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.7743 0.5630 -4.927 8.33e-07 \*\*\*  
## male\_Male 0.3696 0.4340 0.852 0.394   
## race\_Otherwise 0.4420 0.3558 1.242 0.214   
## state\_Kentucky 0.4992 0.4765 1.047 0.295   
## state\_Louisiana 0.6783 0.4623 1.467 0.142   
## state\_Virginia -2.9170 0.5742 -5.080 3.78e-07 \*\*\*  
## crime\_Larceny 0.3503 0.4696 0.746 0.456   
## crime\_Drugs -0.3914 0.4061 -0.964 0.335   
## crime\_Driving -0.2431 0.5516 -0.441 0.659   
## multiple.offenses\_MultipleOffenses 1.5994 0.3692 4.332 1.48e-05 \*\*\*  
## ---  
## 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: 275.30 on 497 degrees of freedom  
## AIC: 295.3  
##   
## Number of Fisher Scoring iterations: 6

### From the above model, you can see that the state of Virginia is once again significant in predicting whether an individual will violate their parole. Another variable that is significant in predicting this is if an individual has committed multiple offenses then they are more likely to violate their parole as well. All of the other variable’s p-values are not significant in predicting parole violation in this model. The AIC for this model is 295.3, which has decreased from the previous models once again, which means that the quality is higher for this model as well. I would say that this model is not very intuitive, only due to the lack of strong predictions that can be made from the above model. I can only predict that a parolee from Virginia is less likely to break their parole, and that someone with multiple offenses is more likely, but nothing beyond that. We need more information to make solid predictions.

# Create a logistic regression model using the training set to predict “violator” using the variables - state, multiple.offenses, and race

selecttrain\_model =   
 logistic\_reg() %>%  
 set\_engine("glm")  
  
selecttrain\_recipe = recipe(violator ~ state + multiple.offenses + race, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
selecttrainlogreg\_wf = workflow() %>%  
 add\_recipe(selecttrain\_recipe) %>%  
 add\_model(selecttrain\_model)  
  
selecttrain\_fit = fit(selecttrainlogreg\_wf, train)

summary(selecttrain\_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\_MultipleOffenses 1.5998 0.3684 4.342 1.41e-05 \*\*\*  
## race\_Otherwise 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 most significant variables in the above model are state and multiple offenses, with Virginia showing to have a less likely prediction of breaking their parole, and individuals with multiple offenses are more likely to break parole. The AIC for this model is 289.99, which is the lowest AIC for any model created so far, leading this model to be the highest quality model.

# Predict probability of breaking parole for Parolee1: Louisiana with multiple offenses and white race

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

## # A tibble: 1 x 2  
## .pred\_NoViolation .pred\_Violation  
## <dbl> <dbl>  
## 1 0.557 0.443

### Based on the above parolee information, this individual has a 55.7% chance of not violating his/her parole, with a 44.3% chance of violating his/her parole.

# Predict probability of breaking parole for Parolee2: Kentucky with no multiple offenses and other race

newdata2 = data.frame(state = "Kentucky", multiple.offenses = "Other", race = "Otherwise")  
predict(selecttrain\_fit, newdata2, type = "prob")

## # A tibble: 1 x 2  
## .pred\_NoViolation .pred\_Violation  
## <dbl> <dbl>  
## 1 0.848 0.152

# The above individual has an 84.8% chance of not violating his/her parole, and a 15.2% chance of violating parole.

predictions = predict(selecttrain\_fit,train,type = "prob")  
head(predictions)

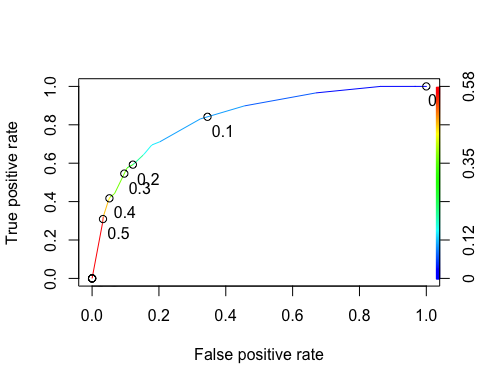
## # A tibble: 6 x 2  
## .pred\_NoViolation .pred\_Violation  
## <dbl> <dbl>  
## 1 0.927 0.0729  
## 2 0.927 0.0729  
## 3 0.927 0.0729  
## 4 0.893 0.107   
## 5 0.893 0.107   
## 6 0.927 0.0729

predictions = predict(selecttrain\_fit,train,type = "prob") [2]  
head(predictions)

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

# Threshold selection

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

# Test thresholds to evaluate accuracy

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

##   
## FALSE TRUE  
## NoViolation 368 80  
## Violation 18 41

## Calculate accuracy

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

## [1] 0.8067061

### The accuracy of the model on the train dataset is 0.8067061 (81% accurate). The sensitivity of the model is 0.7118644, and the specificity of the model is 0.7968750.

### Implications of incorrectly classifying a parolee is that you predict that they will violate parole, and then they do not, or that they will not violate parole, and then they do. I predicted that 80 individuals would not violate parole, and they did violate. I predicted that 18 would violate their parole, and they did not.

# Identify Probability Threshold that Best Maximizes Accuracy

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

##   
## FALSE TRUE  
## NoViolation 434 14  
## Violation 42 17

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

## [1] 0.8895464

t1 = table(train$violator, predictions > 0.4)  
t1

##   
## FALSE TRUE  
## NoViolation 426 22  
## Violation 35 24

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

## [1] 0.887574

t1 = table(train$violator, predictions > 0.3)  
t1

##   
## FALSE TRUE  
## NoViolation 418 30  
## Violation 33 26

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

## [1] 0.8757396

### After trying thresholds of 0.5, 0.4, and 0.3, the threshold with the highest accuracy is 0.5.

# Determine Accuracy of Testing Dataset

test\_model =   
 logistic\_reg() %>%  
 set\_engine("glm")  
  
test\_recipe = recipe(violator ~ state + multiple.offenses + race, test) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
testlogreg\_wf = workflow() %>%  
 add\_recipe(test\_recipe) %>%  
 add\_model(test\_model)  
  
test\_fit = fit(testlogreg\_wf, test)  
  
predictions = predict(test\_fit,test,type = "prob") [2]  
head(predictions)

## # A tibble: 6 x 1  
## .pred\_Violation  
## <dbl>  
## 1 0.339   
## 2 0.0585  
## 3 0.339   
## 4 0.0585  
## 5 0.0585  
## 6 0.0585

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

##   
## FALSE TRUE  
## NoViolation 147 2  
## Violation 9 10

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

## [1] 0.9345238

### The accuracy of the model of the testing dataset is 0.9345238 (93%). The accuracy of the test model at 0.5 as the threshold is higher than the accuracy of the train model, compared at 89%.