# BAN 502 Module 3 - Assign 2

## Kristian Marlowe

### Logistic Regression (Classification)

The first step is to load (and install) necessary programs for the assignment.

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

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✓ ggplot2 3.3.3 ✓ purrr 0.3.4  
## ✓ tibble 3.1.2 ✓ dplyr 1.0.6  
## ✓ tidyr 1.1.3 ✓ stringr 1.4.0  
## ✓ readr 1.4.0 ✓ 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 ──

## ✓ broom 0.7.6 ✓ rsample 0.1.0   
## ✓ dials 0.0.9 ✓ tune 0.1.5   
## ✓ infer 0.5.4 ✓ workflows 0.2.2   
## ✓ modeldata 0.1.0 ✓ workflowsets 0.0.2   
## ✓ parsnip 0.1.5 ✓ yardstick 0.0.8   
## ✓ 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(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

Next, the data will be read-in and variables will be assigned as factors.

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()  
## )

str(parole)

## spec\_tbl\_df [675 × 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()  
## .. )

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

The following code will change variables as factors and recode factors to levels specified in the course assignment.

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, "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4", "Any Other State" = "1")) %>%  
 mutate(crime = as\_factor(crime)) %>% mutate(crime = fct\_recode(crime, "larceny" = "2", "drug-related" = "3", "driving-related" = "4", "other crime" = "1")) %>%  
 mutate(multiple.offenses = as\_factor(multiple.offenses)) %>% mutate(multiple.offenses = fct\_recode(multiple.offenses, "incarcerated for multiple offenses" = "1", "otherwise" = "0")) %>%  
 mutate(violator=as\_factor(violator)) %>% mutate(violator=fct\_recode(violator, "Violate Parole" = "1", "No Parole Violations" ="0"))  
str(parole)

## spec\_tbl\_df [675 × 9] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ male : Factor w/ 2 levels "female","male": 2 1 2 2 2 2 2 1 1 2 ...  
## $ race : Factor w/ 2 levels "white","otherwise": 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 "Any Other State",..: 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 "otherwise","incarcerated for multiple offenses": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "other crime",..: 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : Factor w/ 2 levels "No Parole Violations",..: 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()  
## .. )

summary(parole)

## male race age state   
## female:130 white :389 Min. :18.40 Any Other State:143   
## male :545 otherwise:286 1st Qu.:25.35 Kentucky :120   
## Median :33.70 Louisiana : 82   
## Mean :34.51 Virginia :330   
## 3rd Qu.:42.55   
## Max. :67.00   
## time.served max.sentence multiple.offenses  
## Min. :0.000 Min. : 1.00 otherwise :313   
## 1st Qu.:3.250 1st Qu.:12.00 incarcerated for multiple offenses:362   
## Median :4.400 Median :12.00   
## Mean :4.198 Mean :13.06   
## 3rd Qu.:5.200 3rd Qu.:15.00   
## Max. :6.000 Max. :18.00   
## crime violator   
## other crime :315 No Parole Violations:597   
## larceny :106 Violate Parole : 78   
## drug-related :153   
## driving-related:101   
##   
##

## Task 1

The goal of task 1 is to split the data into training and testing sets.

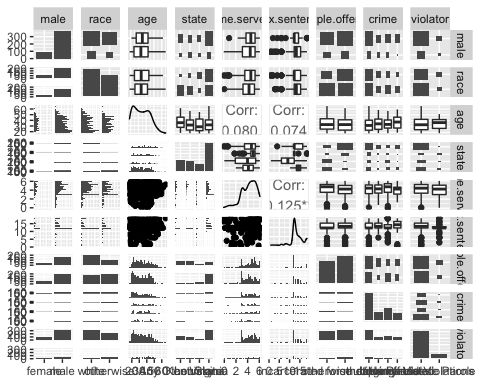
set.seed(12345)  
parole\_split = initial\_split(parole, prop = 0.70, strata = violator)  
train = training(parole\_split)  
test = testing(parole\_split)

## Task 2

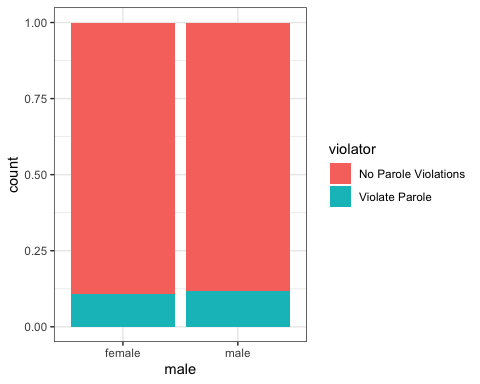
The goal in task 2 is to predict whether or not a parolee will violate his/her parole.

ggpairs(train)

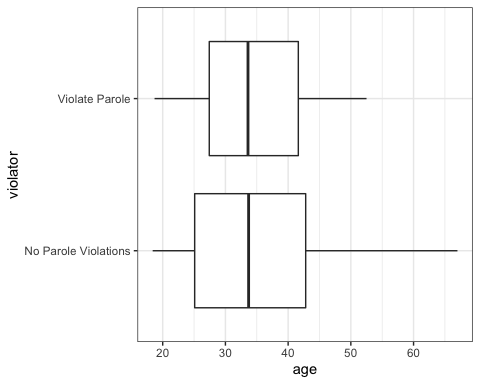
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



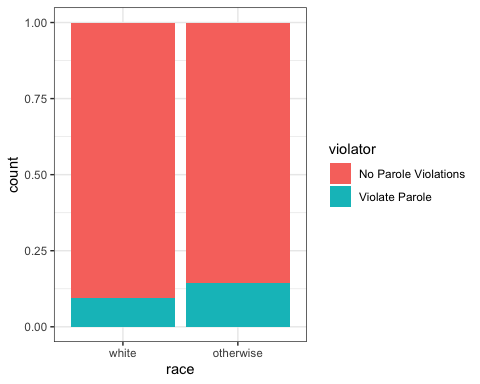
ggplot(parole, aes(x=male, fill = violator)) + geom\_bar(position="fill") + theme\_bw()



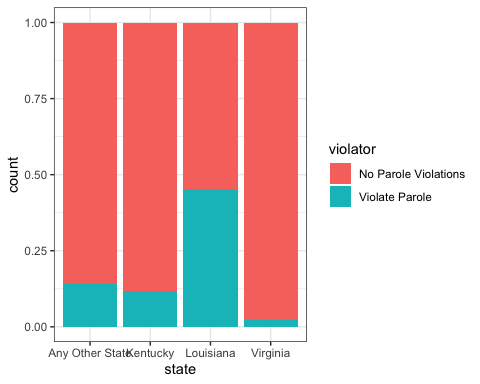
ggplot(parole, aes(x=age, y=violator)) + geom\_boxplot() + theme\_bw()



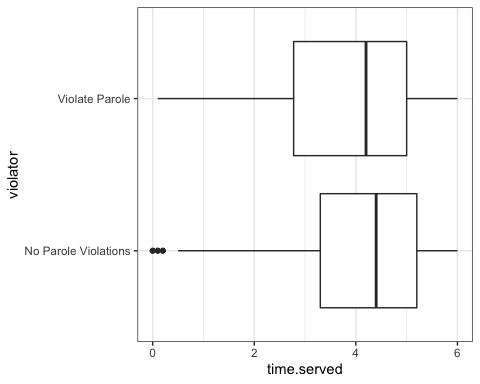
ggplot(parole, aes(x=race, fill = violator)) + geom\_bar(position="fill") + theme\_bw()



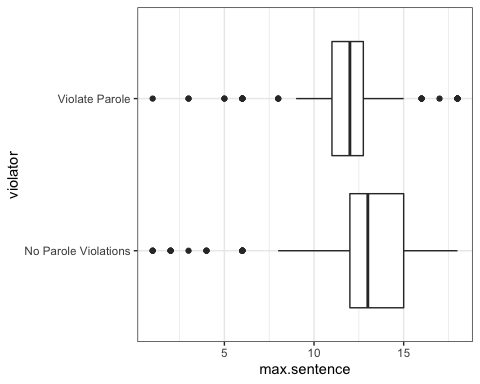
ggplot(parole, aes(x=state, fill = violator)) + geom\_bar(position="fill") + theme\_bw()



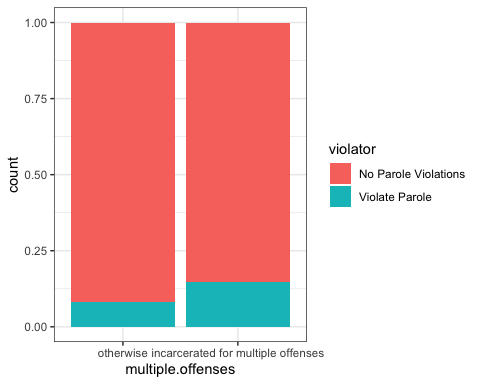
ggplot(parole, aes(x=time.served,y=violator)) + geom\_boxplot() + theme\_bw()



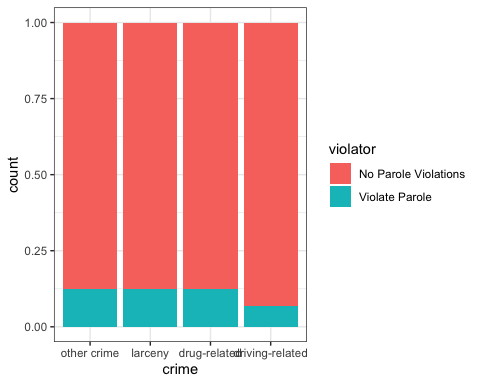
ggplot(parole, aes(x=max.sentence, y=violator)) + geom\_boxplot() + theme\_bw()



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



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



In these visuals, some of the variables that I can see so far that could have an influence on violators would be race, state, max. sentence, crime and multiple.offenses. In race, we can see a slight difference in the percent violating for the ‘otherwise’ race versus the ‘white’. With the state variable, we can see that in Louisiana almost 50% of parolees violate parole again, Virginia has a very low parole violation percentage, and Kentucky and ‘other states’ also have around a 20% chance of breaking parole again. With the variances in percentages this would indicate that where your state is may have something to do with if you violate parole. In the max.sentence boxplot we can see that the median for parole violations is lower than for no parole violations. Although there are a few lower and upper outliers, the spread of the boxplot is smaller as well. This could indicate a slight relationship between the max.sentence and if someone is a violator of their parole. In the multiple.offenses variable, we can see that if someone has been incarcerated for multiple offenses it appears they are about twice as likely to violate parole ( ~6% compared to ~12%). This difference in the percents could indicate a relationship worth looking into. The other variables seem to have fairly equal medians and percents, which would indicate not much of a predictor variable.

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

##   
## female male  
## No Parole Violations 0.8923077 0.8825688  
## Violate Parole 0.1076923 0.1174312

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

##   
## white otherwise  
## No Parole Violations 0.90488432 0.85664336  
## Violate Parole 0.09511568 0.14335664

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

##   
## Any Other State Kentucky Louisiana Virginia  
## No Parole Violations 0.86013986 0.88333333 0.54878049 0.97878788  
## Violate Parole 0.13986014 0.11666667 0.45121951 0.02121212

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

##   
## 1 2 3 4 5  
## No Parole Violations 0.75000000 1.00000000 0.33333333 1.00000000 0.00000000  
## Violate Parole 0.25000000 0.00000000 0.66666667 0.00000000 1.00000000  
##   
## 6 8 9 10 11  
## No Parole Violations 0.56250000 0.85000000 0.66666667 0.66666667 0.60000000  
## Violate Parole 0.43750000 0.15000000 0.33333333 0.33333333 0.40000000  
##   
## 12 13 14 15 16  
## No Parole Violations 0.86496350 0.96103896 0.95454545 0.98333333 0.88888889  
## Violate Parole 0.13503650 0.03896104 0.04545455 0.01666667 0.11111111  
##   
## 17 18  
## No Parole Violations 0.93333333 0.89743590  
## Violate Parole 0.06666667 0.10256410

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

##   
## otherwise incarcerated for multiple offenses  
## No Parole Violations 0.9201278 0.8535912  
## Violate Parole 0.0798722 0.1464088

t6 = table(parole$violator, parole$crime)   
prop.table(t6, margin = 2 )

##   
## other crime larceny drug-related driving-related  
## No Parole Violations 0.87619048 0.87735849 0.87581699 0.93069307  
## Violate Parole 0.12380952 0.12264151 0.12418301 0.06930693

In some instances, we can use the prop.table to further see the changes in our percentages to see if there are good predictors. I chose to exclude the age and time.served variables due to the amount of different decimal values that would be hard to make conclusions from. In this table format, we are able to further see what the graphics show that race, state, max.sentence, and multiple.offenses are some of the greatest predictors we can use based in the changes in the percentages of the violating parole in the different factors. I found it interesting that there was a little more change in the ‘driving-related’ factor of ‘crime’ compared to the other levels, so it may be a variable worth looking at and testing the AOC more. Overall, I would say that state has the greatest influence of changes to the violation of parole with parolees.

## Task 3

The goal of task 3 is to create a logistic regression model with the most predictive variable of violator.

parole\_model =   
 logistic\_reg() %>%   
 set\_engine("glm")   
  
parole\_recipe = recipe(violator ~ state, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())   
  
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   
## -1.0335 -0.5403 -0.2065 -0.2065 2.7780   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.75539 0.28944 -6.065 1.32e-09 \*\*\*  
## state\_Kentucky -0.09521 0.43471 -0.219 0.826636   
## state\_Louisiana 1.40709 0.39351 3.576 0.000349 \*\*\*  
## state\_Virginia -2.08191 0.53672 -3.879 0.000105 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 335.47 on 470 degrees of freedom  
## Residual deviance: 270.95 on 467 degrees of freedom  
## AIC: 278.95  
##   
## Number of Fisher Scoring iterations: 6

In this logistic regression model, we see that three of our factors are significant at an alpha of 0.05 (other states, Louisiana, and Virginia) while Kentucky is not a significant factor. The AIC is 278.95 which could be lowest, however we should test the other variables in order to be sure. Overall this seems to be a fairly decent model, though a little concerning that one variable is not significant.

## Task 4

In this task, the goal is to manually build the best model to predict “violator” using the training data set and use AIC to evaluate the “goodness” of the models.

First I will create a model with all variables in order to compare with our previous model from task 3.

parole\_model2 =   
 logistic\_reg() %>%   
 set\_engine("glm")   
  
parole\_recipe2 = recipe(violator ~ ., train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())   
  
logreg\_wf2 = workflow() %>%  
 add\_recipe(parole\_recipe2) %>%   
 add\_model(parole\_model2)  
  
parole\_fit2 = fit(logreg\_wf2, train)

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

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6354 -0.3931 -0.2624 -0.1370 2.9521   
##   
## Coefficients:  
## Estimate Std. Error  
## (Intercept) -2.959828 1.201122  
## age 0.007477 0.016999  
## time.served -0.099097 0.119169  
## max.sentence 0.066046 0.054472  
## male\_male -0.178372 0.412252  
## race\_otherwise 1.165290 0.405637  
## state\_Kentucky 0.014750 0.501692  
## state\_Louisiana 0.238848 0.555305  
## state\_Virginia -3.771945 0.667998  
## multiple.offenses\_incarcerated.for.multiple.offenses 1.634887 0.398672  
## crime\_larceny 0.412647 0.515017  
## crime\_drug.related -0.151590 0.415229  
## crime\_driving.related -0.717667 0.690246  
## z value Pr(>|z|)   
## (Intercept) -2.464 0.01373 \*   
## age 0.440 0.66004   
## time.served -0.832 0.40565   
## max.sentence 1.212 0.22533   
## male\_male -0.433 0.66525   
## race\_otherwise 2.873 0.00407 \*\*   
## state\_Kentucky 0.029 0.97655   
## state\_Louisiana 0.430 0.66711   
## state\_Virginia -5.647 1.64e-08 \*\*\*  
## multiple.offenses\_incarcerated.for.multiple.offenses 4.101 4.12e-05 \*\*\*  
## crime\_larceny 0.801 0.42300   
## crime\_drug.related -0.365 0.71506   
## crime\_driving.related -1.040 0.29847   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 335.47 on 470 degrees of freedom  
## Residual deviance: 239.68 on 458 degrees of freedom  
## AIC: 265.68  
##   
## Number of Fisher Scoring iterations: 6

parole\_model3 =   
 logistic\_reg() %>%   
 set\_engine("glm")   
  
parole\_recipe3 = recipe(violator ~ race+state+max.sentence+multiple.offenses+crime, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())   
  
logreg\_wf3 = workflow() %>%  
 add\_recipe(parole\_recipe3) %>%   
 add\_model(parole\_model3)  
  
parole\_fit3 = fit(logreg\_wf3, train)

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

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6376 -0.3995 -0.2587 -0.1421 2.8945   
##   
## Coefficients:  
## Estimate Std. Error  
## (Intercept) -3.312971 0.832476  
## max.sentence 0.069489 0.053514  
## race\_otherwise 1.139996 0.400594  
## state\_Kentucky -0.008891 0.488818  
## state\_Louisiana 0.303613 0.536492  
## state\_Virginia -3.771015 0.658425  
## multiple.offenses\_incarcerated.for.multiple.offenses 1.659303 0.396887  
## crime\_larceny 0.413849 0.503626  
## crime\_drug.related -0.138172 0.413535  
## crime\_driving.related -0.664028 0.675176  
## z value Pr(>|z|)   
## (Intercept) -3.980 6.90e-05 \*\*\*  
## max.sentence 1.299 0.19411   
## race\_otherwise 2.846 0.00443 \*\*   
## state\_Kentucky -0.018 0.98549   
## state\_Louisiana 0.566 0.57145   
## state\_Virginia -5.727 1.02e-08 \*\*\*  
## multiple.offenses\_incarcerated.for.multiple.offenses 4.181 2.90e-05 \*\*\*  
## crime\_larceny 0.822 0.41123   
## crime\_drug.related -0.334 0.73829   
## crime\_driving.related -0.983 0.32537   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 335.47 on 470 degrees of freedom  
## Residual deviance: 240.63 on 461 degrees of freedom  
## AIC: 260.63  
##   
## Number of Fisher Scoring iterations: 6

parole\_model4 =   
 logistic\_reg() %>%   
 set\_engine("glm")   
  
parole\_recipe4 = recipe(violator ~ race+state+max.sentence+multiple.offenses, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())   
  
logreg\_wf4 = workflow() %>%  
 add\_recipe(parole\_recipe4) %>%   
 add\_model(parole\_model4)  
  
parole\_fit4 = fit(logreg\_wf4, train)

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

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.4406 -0.3903 -0.2707 -0.1457 2.8826   
##   
## Coefficients:  
## Estimate Std. Error  
## (Intercept) -3.38905 0.81453  
## max.sentence 0.06685 0.05238  
## race\_otherwise 1.11745 0.39552  
## state\_Kentucky 0.05056 0.47336  
## state\_Louisiana 0.36527 0.53807  
## state\_Virginia -3.65744 0.64410  
## multiple.offenses\_incarcerated.for.multiple.offenses 1.70454 0.39860  
## z value Pr(>|z|)   
## (Intercept) -4.161 3.17e-05 \*\*\*  
## max.sentence 1.276 0.20187   
## race\_otherwise 2.825 0.00472 \*\*   
## state\_Kentucky 0.107 0.91494   
## state\_Louisiana 0.679 0.49724   
## state\_Virginia -5.678 1.36e-08 \*\*\*  
## multiple.offenses\_incarcerated.for.multiple.offenses 4.276 1.90e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 335.47 on 470 degrees of freedom  
## Residual deviance: 242.86 on 464 degrees of freedom  
## AIC: 256.86  
##   
## Number of Fisher Scoring iterations: 6

parole\_model5 =   
 logistic\_reg() %>%   
 set\_engine("glm")   
  
parole\_recipe5 = recipe(violator ~ race+state+multiple.offenses, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())   
  
logreg\_wf5 = workflow() %>%  
 add\_recipe(parole\_recipe5) %>%   
 add\_model(parole\_model5)  
  
parole\_fit5 = fit(logreg\_wf5, train)

summary(parole\_fit5$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.3572 -0.4013 -0.2705 -0.1557 2.9726   
##   
## Coefficients:  
## Estimate Std. Error  
## (Intercept) -2.47873 0.36085  
## race\_otherwise 1.11646 0.39092  
## state\_Kentucky -0.01418 0.46926  
## state\_Louisiana 0.11876 0.49950  
## state\_Virginia -3.58422 0.63848  
## multiple.offenses\_incarcerated.for.multiple.offenses 1.65689 0.39652  
## z value Pr(>|z|)   
## (Intercept) -6.869 6.46e-12 \*\*\*  
## race\_otherwise 2.856 0.00429 \*\*   
## state\_Kentucky -0.030 0.97590   
## state\_Louisiana 0.238 0.81206   
## state\_Virginia -5.614 1.98e-08 \*\*\*  
## multiple.offenses\_incarcerated.for.multiple.offenses 4.179 2.93e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 335.47 on 470 degrees of freedom  
## Residual deviance: 244.52 on 465 degrees of freedom  
## AIC: 256.52  
##   
## Number of Fisher Scoring iterations: 6

parole\_model6 =   
 logistic\_reg() %>%   
 set\_engine("glm")   
  
parole\_recipe6 = recipe(violator ~ race+state, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())   
  
logreg\_wf6 = workflow() %>%  
 add\_recipe(parole\_recipe6) %>%   
 add\_model(parole\_model6)  
  
parole\_fit6 = fit(logreg\_wf6, train)

summary(parole\_fit6$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.1700 -0.5274 -0.2421 -0.1448 3.0209   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.9023 0.2998 -6.345 2.22e-10 \*\*\*  
## race\_otherwise 1.0374 0.3690 2.811 0.00494 \*\*   
## state\_Kentucky -0.2349 0.4447 -0.528 0.59730   
## state\_Louisiana 0.8475 0.4466 1.898 0.05773 .   
## state\_Virginia -2.6502 0.5842 -4.536 5.72e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 335.47 on 470 degrees of freedom  
## Residual deviance: 263.02 on 466 degrees of freedom  
## AIC: 273.02  
##   
## Number of Fisher Scoring iterations: 6

From this trial and error of model building, we have a model that includes the variables race, state, and multiple offenses. This model had the lowest AIC of 256.52 compared to the other models. Although another model had a fairly close AIC (race, state, max sentence, multiple offenses) there were more variables significant in the race, state, and multiple offenses model.

## Task 5

In this task, the goal is to create a logistic regression model using the training set to predict “violator” using the variables: state, multiple.offenses, and race.

parole\_model7 =   
 logistic\_reg() %>%   
 set\_engine("glm")   
  
parole\_recipe7 = recipe(violator ~ state + multiple.offenses + race, train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())   
  
logreg\_wf7 = workflow() %>%  
 add\_recipe(parole\_recipe7) %>%   
 add\_model(parole\_model7)  
  
parole\_fit7 = fit(logreg\_wf7, train)

summary(parole\_fit7$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.3572 -0.4013 -0.2705 -0.1557 2.9726   
##   
## Coefficients:  
## Estimate Std. Error  
## (Intercept) -2.47873 0.36085  
## state\_Kentucky -0.01418 0.46926  
## state\_Louisiana 0.11876 0.49950  
## state\_Virginia -3.58422 0.63848  
## multiple.offenses\_incarcerated.for.multiple.offenses 1.65689 0.39652  
## race\_otherwise 1.11646 0.39092  
## z value Pr(>|z|)   
## (Intercept) -6.869 6.46e-12 \*\*\*  
## state\_Kentucky -0.030 0.97590   
## state\_Louisiana 0.238 0.81206   
## state\_Virginia -5.614 1.98e-08 \*\*\*  
## multiple.offenses\_incarcerated.for.multiple.offenses 4.179 2.93e-05 \*\*\*  
## race\_otherwise 2.856 0.00429 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 335.47 on 470 degrees of freedom  
## Residual deviance: 244.52 on 465 degrees of freedom  
## AIC: 256.52  
##   
## Number of Fisher Scoring iterations: 6

This model is a better fit than state alone. The AIC has decreased to 256.52 which is a fairly good drop signifying this is a slightly better model. The variables that are significant are other states, state\_virginia, multiple.offenses\_incarcerated.for.multiple.offenses, and race\_otherwise. The states of Kentucky and Louisiana are not significant in this model. The estimated values also ‘make sense’ in this instance. There are positive values for the estimates of ‘other race’, ‘living in Louisiana’ and ‘multiple offenses’ which we saw was a predictor from the visualizations. The other values are negative meaning it lowers your chances of violating parole.

## Task 6

The goal of this task is to find the predicted probability of parole violation of two parolees.

parolee1 = data.frame(state= "Louisiana", multiple.offenses = "incarcerated for multiple offenses", race = "white")  
predict(parole\_fit7, parolee1, type="prob")

## # A tibble: 1 x 2  
## `.pred\_No Parole Violations` `.pred\_Violate Parole`  
## <dbl> <dbl>  
## 1 0.669 0.331

The predicted parole violations for Parolee 1: from Louisiana with multiple offenses and of the white race is 0.3311299 or ~33.11%.

parolee2 = data.frame(state= "Kentucky", multiple.offenses = "otherwise", race = "otherwise")  
predict(parole\_fit7, parolee2, type="prob")

## # A tibble: 1 x 2  
## `.pred\_No Parole Violations` `.pred\_Violate Parole`  
## <dbl> <dbl>  
## 1 0.798 0.202

The predicted parole violations for Parolee 2: from Kentucky with no multiple offenses and other race are 0.2015788 or ~ 20.12%.

## Task 7

In this task, the goal is to develop an ROC curve and determine the probability threshold that best balances specificity and sensitivity on the training set.

This is creation of predictions variable.

predictions = predict(parole\_fit7, train, type="prob")  
head(predictions)

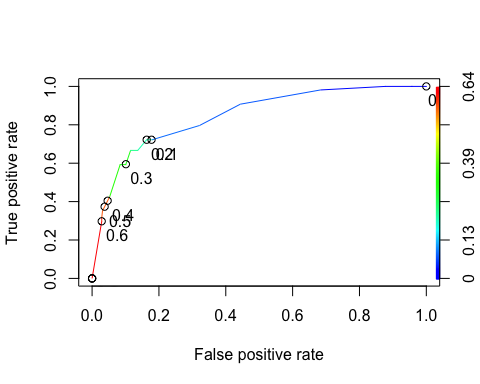
## # A tibble: 6 x 2  
## `.pred\_No Parole Violations` `.pred\_Violate Parole`  
## <dbl> <dbl>  
## 1 0.923 0.0774  
## 2 0.796 0.204   
## 3 0.796 0.204   
## 4 0.923 0.0774  
## 5 0.923 0.0774  
## 6 0.796 0.204

predictions = predict(parole\_fit7, train, type="prob")[2]  
head(predictions)

## # A tibble: 6 x 1  
## `.pred\_Violate Parole`  
## <dbl>  
## 1 0.0774  
## 2 0.204   
## 3 0.204   
## 4 0.0774  
## 5 0.0774  
## 6 0.204

This is the 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))



Area under the curve selection.

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

## [1] 0.8460121

The value of 0.8460121 means that this is close to 1, meaning that it is a fairly good AOC value.

The code below will determine the threshold for sensitivity and selectivity.

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 0.7222222  
## specificity 0.8369305  
## cutoff 0.2015788

The ideal cutoff to find the ideal threshold to get as close to balanced as possible would be 0.2015788. This would give us a sensitivity of 0.7222222 and a specificity of 0.8369305.

## Task 8

This task will find the accuracy, sensitivity, and specificity of the model on the training set given the cutoff from Task 7. It will also detail the implications of incorrectly classifying a parolee.

t7= table(train$violator, predictions > 0.2015788)  
t7

##   
## FALSE TRUE  
## No Parole Violations 360 57  
## Violate Parole 18 36

This will test the accuracy.

(t7[1,1] + t7[2,2]) / nrow(train)

## [1] 0.8407643

This will calculate the sensitivity

36 / (36+18)

## [1] 0.6666667

This will calculate specificity.

360/(360+57)

## [1] 0.8633094

The accuracy was 0.8407643, the sensitivity was 0.6666667, and the specificity was 0.8633094.

The implications of incorrectly classifying a parolee are important. We would not like to predict someone would not violate their parole and then they do, which could potentially cause harm to other citizens or the parolee depending on the type of violation. We are discussing someone’s livelihood and freedom in this instance so being ‘right’ in our model is not the best approach. I would much rather be wrong and predict someone would violate their parole and then find they do not violate parole and allow them to continue to be free.

## Task 9

This task’s goal is to: Identify a probability threshold (via trial-and-error) that best maximizes accuracy on the training set.

Threshold = 0.5

t8 = table(train$violator,predictions > 0.5)  
t8

##   
## FALSE TRUE  
## No Parole Violations 404 13  
## Violate Parole 35 19

(t8[1,1]+t8[2,2])/nrow(train)

## [1] 0.8980892

Threshold = 0.6

t9 = table(train$violator,predictions > 0.6)  
t9

##   
## FALSE TRUE  
## No Parole Violations 405 12  
## Violate Parole 38 16

(t9[1,1]+t9[2,2])/nrow(train)

## [1] 0.8938429

t10 = table(train$violator,predictions > 1)   
t10

##   
## FALSE  
## No Parole Violations 417  
## Violate Parole 54

(t10[1])/nrow(train)

## [1] 0.8853503

The probability threshold that maximizes the accuracy on the training set was 0.5. This created an accuracy of 0.8980892 which was slightly higher than the threshold of 0.6 and 1. I also tried the other decimals from 0-1 in tenths increments, and these were the top three accuracy levels. This is a little bit of lopsided data in comparison of no parole violations and parole violations, so one could argue that you could choose the probability threshold of 1.0. However, in my opinion in this instance when discussing parole violations and the potential impacts that it could bring on the life of the parolee and potentially the lives of others, I would argue that having that extra ~1% is important.

## Task 10

This task’s goal is to: Use the probability threshold from Task 9 to determine the accuracy of the model on the testing set.

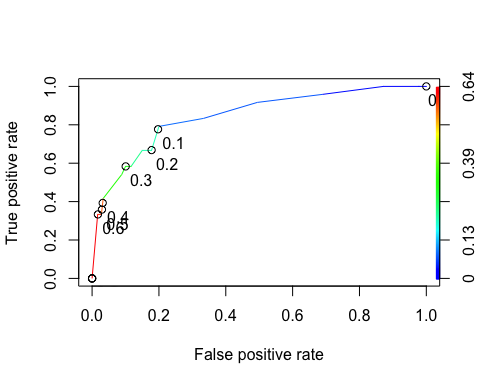
predictions2 = predict(parole\_fit7, test, type="prob")  
head(predictions2)

## # A tibble: 6 x 2  
## `.pred\_No Parole Violations` `.pred\_Violate Parole`  
## <dbl> <dbl>  
## 1 0.923 0.0774  
## 2 0.923 0.0774  
## 3 0.796 0.204   
## 4 0.923 0.0774  
## 5 0.427 0.573   
## 6 0.796 0.204

predictions2 = predict(parole\_fit7, test, type="prob")[2]  
head(predictions2)

## # A tibble: 6 x 1  
## `.pred\_Violate Parole`  
## <dbl>  
## 1 0.0774  
## 2 0.0774  
## 3 0.204   
## 4 0.0774  
## 5 0.573   
## 6 0.204

ROCRpred2 = prediction(predictions2, test$violator)  
  
ROCRperf2 = performance(ROCRpred2, "tpr", "fpr")  
plot(ROCRperf2, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))



as.numeric(performance(ROCRpred2, "auc")@y.values)

## [1] 0.8456019

t11= table(test$violator, predictions2 > 0.2015788)  
t11

##   
## FALSE TRUE  
## No Parole Violations 153 27  
## Violate Parole 8 16

The following will test the accuracy

(t11[1,1]+t11[2,2])/nrow(test)

## [1] 0.8284314

The following will calculate the sensitivity

16/(16+8)

## [1] 0.6666667

The following will calculate the specificity

153/(153+27)

## [1] 0.85

Now we will do trial and error to maximize the accuracy of the accuracy test.

t12 = table(test$violator,predictions2 > 0.5)  
t12

##   
## FALSE TRUE  
## No Parole Violations 175 5  
## Violate Parole 16 8

(t12[1,1]+t12[2,2])/nrow(test)

## [1] 0.8970588

t13 = table(test$violator,predictions2 > 0.6)  
t13

##   
## FALSE TRUE  
## No Parole Violations 177 3  
## Violate Parole 16 8

(t13[1,1]+t13[2,2])/nrow(test)

## [1] 0.9068627

t14 = table(test$violator,predictions2 > 1)  
t14

##   
## FALSE  
## No Parole Violations 180  
## Violate Parole 24

(t14[1])/nrow(test)

## [1] 0.8823529

When using the probability threshold from Task 9 (0.1295001) to test the accuracy of the model on the testing set, we see that using the threshold >0.6 we find the greatest accuracy on the model at 0.9.