Mod3Assign2

Adwoa Kessie

2/6/2022

library(e1071)

## Warning: package 'e1071' was built under R version 4.1.2

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.1.2

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

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.5 v dplyr 1.0.7  
## v tidyr 1.1.4 v stringr 1.4.0  
## v readr 2.0.2 v forcats 0.5.1

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

library(tidymodels)

## Warning: package 'tidymodels' was built under R version 4.1.2

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

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

## v broom 0.7.9 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

## Warning: package 'dials' was built under R version 4.1.2

## Warning: package 'infer' was built under R version 4.1.2

## Warning: package 'modeldata' was built under R version 4.1.2

## Warning: package 'parsnip' was built under R version 4.1.2

## Warning: package 'recipes' was built under R version 4.1.2

## Warning: package 'rsample' was built under R version 4.1.2

## Warning: package 'tune' was built under R version 4.1.2

## Warning: package 'workflows' was built under R version 4.1.2

## Warning: package 'workflowsets' was built under R version 4.1.2

## Warning: package 'yardstick' was built under R version 4.1.2

## -- 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 rsample::permutations() masks e1071::permutations()  
## x yardstick::spec() masks readr::spec()  
## x recipes::step() masks stats::step()  
## x tune::tune() masks e1071::tune()  
## \* Search for functions across packages at https://www.tidymodels.org/find/

library(ROCR) #for Lasso, ridge, and elastic net models

## Warning: package 'ROCR' was built under R version 4.1.2

parole = read\_csv("parole.csv")

## Rows: 675 Columns: 9

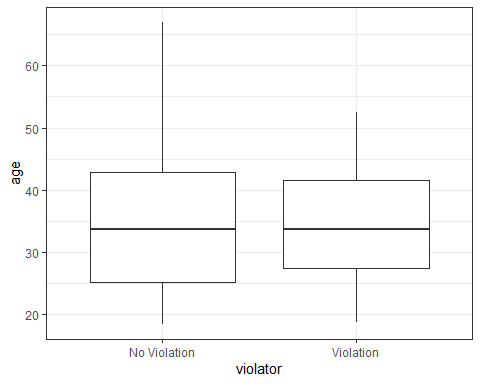
## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## dbl (9): male, race, age, state, time.served, max.sentence, multiple.offense...

##   
## 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.

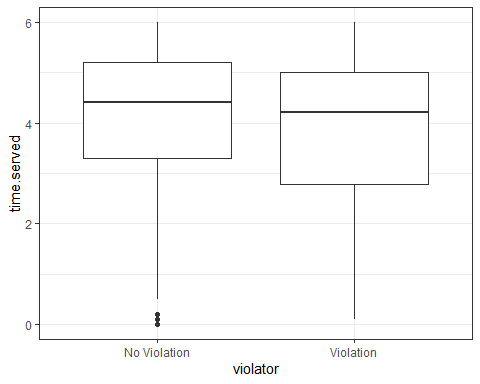
parole = parole %>% mutate(male = as\_factor(male)) %>% mutate(male = fct\_recode(male, "Female" = "0", "Male" = "1" ))  
parole = parole %>% mutate(race = as\_factor(race)) %>% mutate(race = fct\_recode(race, "Others" = "2", "White" = "1" ))  
parole = parole %>% mutate(state = as\_factor(state)) %>% mutate(state = fct\_recode(state, "Other" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4"))  
parole = parole %>% mutate(multiple.offenses = as\_factor(multiple.offenses)) %>% mutate(multiple.offenses = fct\_recode(multiple.offenses, "Other" = "0", "Mulitple" = "1" ))  
parole = parole %>% mutate(violator = as\_factor(violator)) %>% mutate(violator = fct\_recode(violator, "No Violation" = "0", "Violation" = "1" ))  
parole = parole %>% mutate(crime = as\_factor(crime)) %>% mutate(crime = fct\_recode(crime, "Larceny" = "2", "Drugs" = "3", "Other" = "1", "Driving" = "4" ))

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

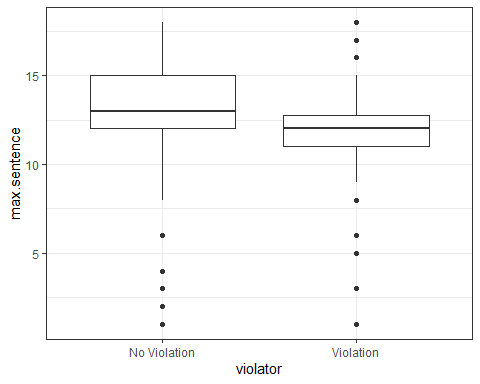
ggplot(parole,aes(x=violator,y=age)) + geom\_boxplot() + theme\_bw()



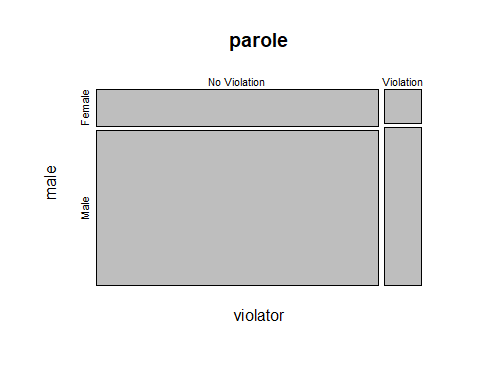
ggplot(parole,aes(x=violator,y=time.served)) + geom\_boxplot() + theme\_bw()



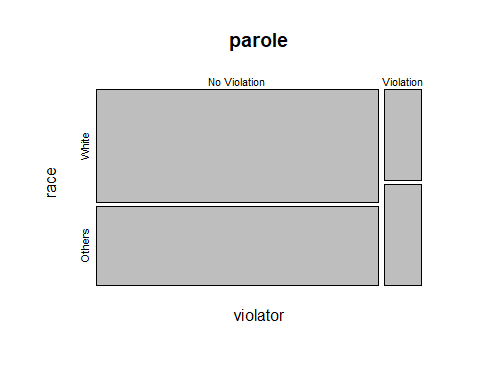
ggplot(parole,aes(x=violator,y=max.sentence)) + geom\_boxplot() + theme\_bw()



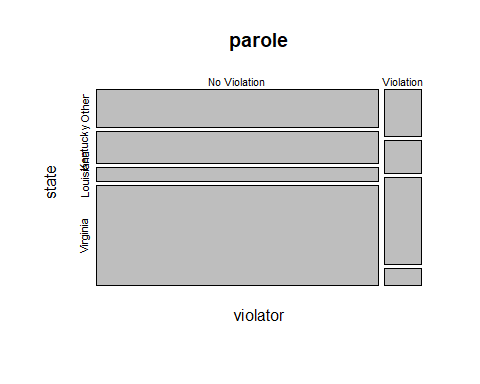
mosaicplot(violator~male,data=parole)



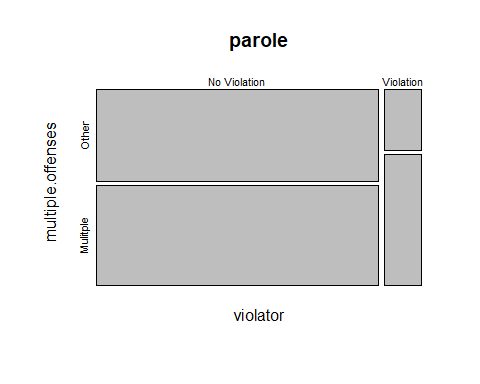
mosaicplot(violator~race,data=parole)



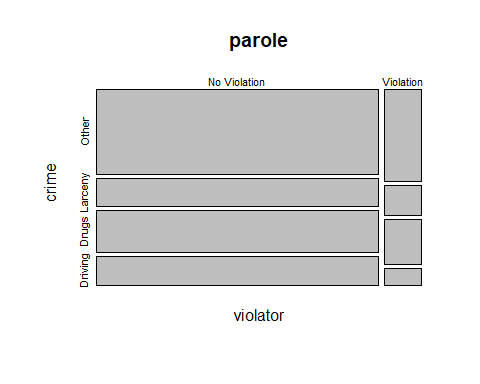
mosaicplot(violator~state,data=parole)



mosaicplot(violator~multiple.offenses,data=parole)



mosaicplot(violator~crime,data=parole)



model = glm(violator~state, family="binomial", data = train)  
summary(model)

##   
## Call:  
## glm(formula = violator ~ state, family = "binomial", data = train)  
##   
## 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 \*\*\*  
## stateKentucky -0.09521 0.43471 -0.219 0.826636   
## stateLouisiana 1.40709 0.39351 3.576 0.000349 \*\*\*  
## stateVirginia -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

## State has a significant impact on Violation. Loisiana and Virginia have p-values that are preetty significant.

model = glm(violator~state+multiple.offenses, family="binomial", data = train)  
summary(model)

##   
## Call:  
## glm(formula = violator ~ state + multiple.offenses, family = "binomial",   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.1999 -0.4326 -0.2271 -0.2271 2.7095   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.3218 0.3519 -6.598 4.17e-11 \*\*\*  
## stateKentucky 0.1620 0.4583 0.353 0.7237   
## stateLouisiana 0.7652 0.4335 1.765 0.0775 .   
## stateVirginia -2.9325 0.5755 -5.095 3.48e-07 \*\*\*  
## multiple.offensesMulitple 1.6093 0.3875 4.153 3.29e-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: 252.79 on 466 degrees of freedom  
## AIC: 262.79  
##   
## Number of Fisher Scoring iterations: 6

## The final model uses state and multiple.offenses parameters to predict violation. The AIC of this model is better than all other models I tried and also the one I created previously that only contained state and independent variable.

model = glm(violator~state+multiple.offenses+race, family="binomial", data = train)  
summary(model)

##   
## Call:  
## glm(formula = violator ~ state + multiple.offenses + race, family = "binomial",   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.3572 -0.4013 -0.2705 -0.1557 2.9726   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.47873 0.36085 -6.869 6.46e-12 \*\*\*  
## stateKentucky -0.01418 0.46926 -0.030 0.97590   
## stateLouisiana 0.11876 0.49950 0.238 0.81206   
## stateVirginia -3.58422 0.63848 -5.614 1.98e-08 \*\*\*  
## multiple.offensesMulitple 1.65689 0.39652 4.179 2.93e-05 \*\*\*  
## raceOthers 1.11646 0.39092 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 turned about to be more efficient than my best model. The aIC for this model is 256.52 vs. my model which was 262.79. All the three variables are pretty significant. However, race is the least significant.

# Parolee 1

parolee1 = -2.47873+0.11876+1.65689  
print(parolee1)

## [1] -0.70308

# Parolee 2

parolee2 = -2.47873-0.01418+1.11646  
print(parolee2)

## [1] -1.37645

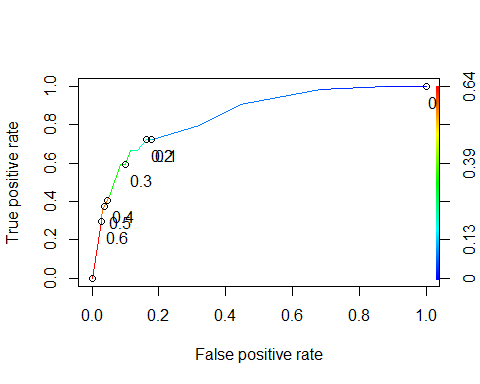
Let’s extract just the “Yes” prediction.

predictions = predict(model, train, type="response")  
head(predictions)

## 1 2 3 4 5 6   
## 0.07736254 0.20387040 0.20387040 0.07736254 0.07736254 0.20387040

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] 0.8460121

#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 0.7222222  
## specificity 0.8369305  
## cutoff 0.2015788

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 > 0.2015788)  
t1

##   
## FALSE TRUE  
## No Violation 360 57  
## Violation 18 36

Calculate accuracy

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

## [1] 0.8407643

Sensitivity

36/(36+18)

## [1] 0.6666667

Specificity

360/(360+57)

## [1] 0.8633094

## Incorrectly classifying a parolee as voilatated can lead to severe impact on the person. The person who has a chance to get out early on parole, might lose out on the chance.

Can 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 404 13  
## Violation 35 19

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

## [1] 0.8980892

Threshold = 0.6

t1 = table(train$violator,predictions > 0.6)  
t1

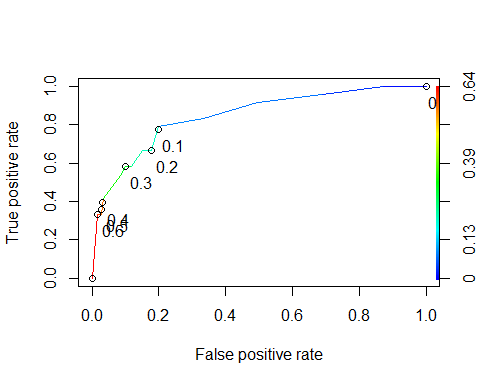
##   
## FALSE TRUE  
## No Violation 405 12  
## Violation 38 16

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

## [1] 0.8938429

predictions = predict(model, test, type="response")

#Change this next line to the names of your predictions and the response variable in the training data frame  
ROCRpred = prediction(predictions, test$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] 0.8456019

#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 0.79166667  
## specificity 0.80000000  
## cutoff 0.08627651

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

##   
## FALSE TRUE  
## No Violation 175 5  
## Violation 16 8

Calculate accuracy

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

## [1] 0.388535