LogRegAssign

Christina Rakes

9/12/2021

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, "No" = "0", "Yes" = "1" ))  
  
parole = parole %>% mutate(race = as\_factor(race)) %>%   
 mutate(race = fct\_recode(race, "Other" = "2", "white" = "1" ))  
  
parole = parole %>% mutate(state = as\_factor(state)) %>%   
 mutate(state = fct\_recode(state, "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4", "Other" = "1"))  
  
parole = parole %>% mutate(crime = as\_factor(crime)) %>%   
 mutate(crime = fct\_recode(crime, "larceny" = "2", "drug-related crime" = "3", "driving-related crime" = "4", "other" = "1" ))  
  
parole = parole %>% mutate(multiple.offenses = as\_factor(multiple.offenses)) %>%   
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "single" = "0", "multiple" = "1" ))  
  
parole = parole %>% mutate( violator= as\_factor(violator)) %>%   
 mutate(violator = fct\_recode(violator, "no violation" = "0", "violated" = "1" ))

set.seed(12345)  
parole\_split = initial\_split(parole, prob = .70, strata = violator)  
train = training(parole\_split)  
test = testing(parole\_split)

parole\_model =  
 logistic\_reg()%>%  
 set\_engine("glm")  
  
parole\_recipe = recipe(violator ~., train)%>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
parole\_wflow =   
 workflow()%>%  
 add\_recipe(parole\_recipe)%>%  
 add\_model(parole\_model)  
  
parole\_fit = fit(parole\_wflow, train)  
  
summary(parole\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6556 -0.4192 -0.2245 -0.1334 3.0796   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.19534 1.17870 -2.711 0.00671 \*\*   
## age 0.01402 0.01621 0.864 0.38732   
## time.served -0.09637 0.11064 -0.871 0.38372   
## max.sentence 0.03553 0.05001 0.710 0.47743   
## male\_Yes 0.53446 0.46672 1.145 0.25215   
## race\_Other 0.82407 0.38321 2.150 0.03152 \*   
## state\_Kentucky -0.29537 0.49041 -0.602 0.54698   
## state\_Louisiana 0.41773 0.52216 0.800 0.42371   
## state\_Virginia -3.80938 0.66750 -5.707 1.15e-08 \*\*\*  
## multiple.offenses\_multiple 1.52162 0.38001 4.004 6.22e-05 \*\*\*  
## crime\_larceny 0.31535 0.50623 0.623 0.53332   
## crime\_drug.related.crime -0.16365 0.41626 -0.393 0.69420   
## crime\_driving.related.crime -0.50443 0.61921 -0.815 0.41528   
## ---  
## 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: 252.63 on 492 degrees of freedom  
## AIC: 278.63  
##   
## Number of Fisher Scoring iterations: 7

It looks like “multiple.offenses seems to be a good predictor of”violator". This makes sense to me that people who are repeat offenders would be more likely to violate parole.

parole2\_model =  
 logistic\_reg()%>%  
 set\_engine("glm")  
  
parole2\_recipe = recipe(violator ~ multiple.offenses, train)%>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
parole2\_wflow =   
 workflow()%>%  
 add\_recipe(parole2\_recipe)%>%  
 add\_model(parole2\_model)  
  
parole2\_fit = fit(parole2\_wflow, train)  
  
summary(parole2\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.5564 -0.5564 -0.4125 -0.4125 2.2390   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.4215 0.2394 -10.116 <2e-16 \*\*\*  
## multiple.offenses\_multiple 0.6341 0.2954 2.147 0.0318 \*   
## ---  
## 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: 355.27 on 503 degrees of freedom  
## AIC: 359.27  
##   
## Number of Fisher Scoring iterations: 5

After running the logistic regression the multiple.offenses variable doesn’t seem to look as strong as it did for predicting violator?

parole2\_model =  
 logistic\_reg()%>%  
 set\_engine("glm")  
  
parole2\_recipe = recipe(violator ~ state, train)%>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
parole2\_wflow =   
 workflow()%>%  
 add\_recipe(parole2\_recipe)%>%  
 add\_model(parole2\_model)  
  
parole2\_fit = fit(parole2\_wflow, train)  
  
summary(parole2\_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 \*\*\*  
## state\_Kentucky -0.5281 0.4404 -1.199 0.230480   
## state\_Louisiana 1.3928 0.3656 3.809 0.000139 \*\*\*  
## state\_Virginia -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

parole2\_model =  
 logistic\_reg()%>%  
 set\_engine("glm")  
  
parole2\_recipe = recipe(violator ~ race, train)%>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
parole2\_wflow =   
 workflow()%>%  
 add\_recipe(parole2\_recipe)%>%  
 add\_model(parole2\_model)  
  
parole2\_fit = fit(parole2\_wflow, train)  
  
summary(parole2\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.5496 -0.5496 -0.4498 -0.4498 2.1639   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.2399 0.1988 -11.269 <2e-16 \*\*\*  
## race\_Other 0.4262 0.2798 1.523 0.128   
## ---  
## 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: 357.79 on 503 degrees of freedom  
## AIC: 361.79  
##   
## Number of Fisher Scoring iterations: 4

parole2\_model =  
 logistic\_reg()%>%  
 set\_engine("glm")  
  
parole2\_recipe = recipe(violator ~ crime, train)%>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
parole2\_wflow =   
 workflow()%>%  
 add\_recipe(parole2\_recipe)%>%  
 add\_model(parole2\_model)  
  
parole2\_fit = fit(parole2\_wflow, train)  
  
summary(parole2\_fit$fit$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 \*\*\*  
## crime\_larceny -0.3017 0.4181 -0.721 0.471   
## crime\_drug.related.crime -0.0401 0.3431 -0.117 0.907   
## crime\_driving.related.crime -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

parole2\_model =  
 logistic\_reg()%>%  
 set\_engine("glm")  
  
parole2\_recipe = recipe(violator ~ male, train)%>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
parole2\_wflow =   
 workflow()%>%  
 add\_recipe(parole2\_recipe)%>%  
 add\_model(parole2\_model)  
  
parole2\_fit = fit(parole2\_wflow, train)  
  
summary(parole2\_fit$fit$fit$fit)

##   
## Call:  
## stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.5107 -0.5107 -0.5107 -0.4172 2.2293   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.3979 0.3693 -6.494 8.38e-11 \*\*\*  
## male\_Yes 0.4266 0.3989 1.069 0.285   
## ---  
## 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: 358.86 on 503 degrees of freedom  
## AIC: 362.86  
##   
## Number of Fisher Scoring iterations: 5

The best model looks to be with “state”. The AIC is lower than all other models ran and shows great significance in predicting “violator” in Virginia and Louisiana.

parole2\_model =  
 logistic\_reg()%>%  
 set\_engine("glm")  
  
parole2\_recipe = recipe(violator ~ state + multiple.offenses + race, train)%>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
parole2\_wflow =   
 workflow()%>%  
 add\_recipe(parole2\_recipe)%>%  
 add\_model(parole2\_model)  
  
parole2\_fit = fit(parole2\_wflow, train)  
  
summary(parole2\_fit$fit$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 2.75e-12 \*\*\*  
## state\_Kentucky -0.4641 0.4627 -1.003 0.3159   
## state\_Louisiana 0.3328 0.4577 0.727 0.4671   
## state\_Virginia -3.7640 0.6526 -5.768 8.03e-09 \*\*\*  
## multiple.offenses\_multiple 1.5128 0.3704 4.084 4.43e-05 \*\*\*  
## race\_Other 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

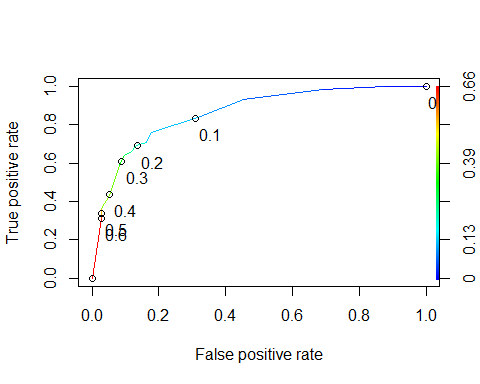
Based on the AIC the quality of this model appears to be good with a lower AIC than any of the previous models. This model shows the greatest significance with multiple.offenses and the state of Virginia.

newdata1 = data.frame(state = "Louisiana", multiple.offenses = "multiple", race = "white")  
newdata2 = data.frame(state = "Kentucky", multiple.offenses = "single", race = "other")  
predict(parole2\_fit, newdata1, type = "prob")

## # A tibble: 1 x 2  
## `.pred\_no violation` .pred\_violated  
## <dbl> <dbl>  
## 1 0.597 0.403

predictions = predict(parole2\_fit, train, type = "prob")[2]

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] 0.8606804

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

t1 = table(train$violator,predictions > 0.1295001)  
t1

##   
## FALSE TRUE  
## no violation 374 73  
## violated 17 41

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

## [1] 0.8217822

41/(17+41)

## [1] 0.7068966

374/(374+73)

## [1] 0.836689

What are the implications of incorrectly classifying a parolee? Being classified incorrectly could cause someone to overlook someone who could violate parole.

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

##   
## FALSE TRUE  
## no violation 435 12  
## violated 39 19

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

## [1] 0.8990099

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

##   
## FALSE TRUE  
## no violation 424 23  
## violated 33 25

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

## [1] 0.8891089

Task 9 - .5 seems to have the best accuracy.

Task 10 - .8990099 Accuracy