# Classification with Logistic Regression

### Miguel Fernandez

### BAN-502

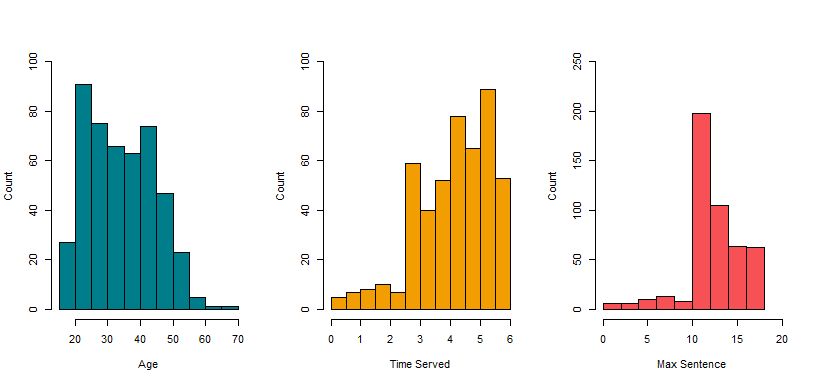
# Import libraries  
library(tidyverse)  
library(MASS)  
library(caret)  
library(ROCR)  
library(GGally)  
library(gridExtra)  
  
# Set global theme  
theme\_set(theme\_bw())  
  
# Read in data  
parole <- read\_csv("parole.csv")  
  
# Recode variables and levels  
parole <- parole %>%  
 mutate(male = as\_factor(as.character(male))) %>%  
 mutate(male = fct\_recode(male,"male" = "1", "female" = "0")) %>%  
 mutate(race = as\_factor(as.character(race))) %>%  
 mutate(race = fct\_recode(race,"white" = "1", "not.white" = "2")) %>%  
 mutate(state = as\_factor(as.character(state))) %>%  
 mutate(state = fct\_recode(state,"ky" = "2",  
 "la" = "3",  
 "va" = "4",  
 "other" = "1")) %>%  
 mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses,"yes" = "1",  
 "no" = "0")) %>%  
 mutate(crime = as\_factor(as.character(crime))) %>%  
 mutate(crime = fct\_recode(crime,"larceny" = "2",  
 "drug" = "3",  
 "driving" = "4",  
 "other" = "1")) %>%  
 mutate(violator = as\_factor(as.character(violator))) %>%  
 mutate(violator = fct\_recode(violator,"yes" = "1", "no" = "0"))

#### Task 1

# Set # random seed  
set.seed(12345)  
# Split data for training and testing  
train.rows <- createDataPartition(y=parole$violator,  
 p=0.7,  
 list = FALSE)  
train <- slice(parole, train.rows)  
test <- slice(parole, -train.rows)

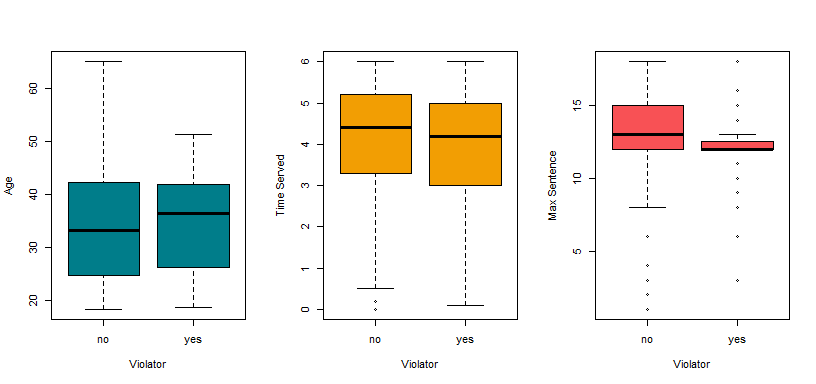
#### Task 2

# Check distribution of numeric variables  
par(mfrow = c(1,3))  
  
hist(train$age, ylab="Count", xlab= "Age",  
 main = NULL, col="#007D8A", ylim = c(0, 100))  
hist(train$time.served, ylab="Count", xlab= "Time Served",  
 main = NULL, col="#F29E03", ylim = c(0, 100))  
hist(train$max.sentence, ylab="Count", xlab= "Max Sentence",  
 main = NULL, col="#F85155", ylim = c(0, 250), xlim = c(0,20))



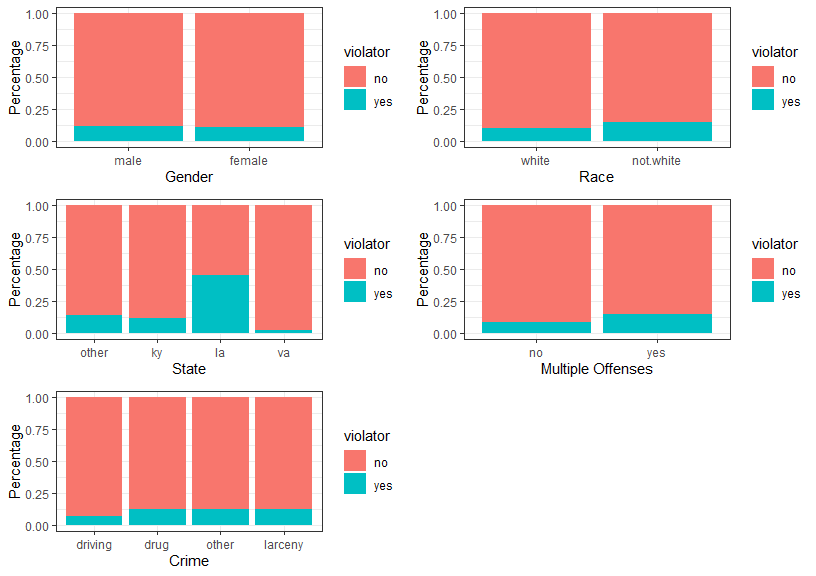
The histograms above reveal skewed distributions across the three numeric variables. The train data set contains younger parolees, mostly between 20 and 45 years old. There are very few observations greater than 60 years old. Time Served and Max Sentence are negatively skewed with a majority of observations towards the maximum values in the data set. Let’s see how the variables are distributed across the response variable, violator.

# Check distribution of numeric variables against violator  
par(mfrow = c(1,3))  
  
boxplot(age~violator, ylab="Age", xlab= "Violator",  
 col="#007D8A",data = train)  
boxplot(time.served~violator, ylab="Time Served", xlab= "Violator",  
 col="#F29E03",data = train)  
boxplot(max.sentence~violator, ylab="Max Sentence", xlab= "Violator",  
 col="#F85155",data = train)



The boxplots above uncover interesting patterns. Age appears signifcant as older parolees tend to violate parole. Those who served more time appear to violate parole less. Perhaps they left the system reformed? And those whose maximum sentences were largest, violated parole less.

# Create barplots for factored variables  
p1 <- ggplot(parole, aes(x = male, fill = violator)) +  
 geom\_bar(position = "fill") +  
 ylab("Percentage") +  
 xlab("Gender")  
p2 <- ggplot(parole, aes(x = race, fill = violator)) +  
 geom\_bar(position = "fill") +  
 ylab("Percentage") +  
 xlab("Race")  
p3 <- ggplot(parole, aes(x = state, fill = violator)) +  
 geom\_bar(position = "fill") +  
 ylab("Percentage") +  
 xlab("State")  
p4 <- ggplot(parole, aes(x = multiple.offenses, fill = violator)) +  
 geom\_bar(position = "fill") +  
 ylab("Percentage") +  
 xlab("Multiple Offenses")  
p5 <- ggplot(parole, aes(x = crime, fill = violator)) +  
 geom\_bar(position = "fill") +  
 ylab("Percentage") +  
 xlab("Crime")  
  
grid.arrange(p1, p2, p3, p4, p5, ncol = 2)



The barplots above reveal that a significant portion of observations have not committed parole violations. However, we can see some patterns that may help a model determine whether or not a person will violate parole. For example, Louisana contains the largest number of violators. The plot reveals that both genders are equally as likely to violate parole so by itself, male may not be a strong predictor of parole violation. Somewhat intuitively, we can see that those convicted of multiple offenses are more likely to violate parole and those with driving crimes, which can be less offensive, are less likely to violate parole.

#### Task 3

# Construct model with state and violator  
mod1 <- glm(violator ~ state, train, family = "binomial")  
summary(mod1)

##   
## Call:  
## glm(formula = violator ~ state, family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.0335 -0.5589 -0.2065 -0.2065 2.7780   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.75539 0.28944 -6.065 1.32e-09 \*\*\*  
## stateky -0.02238 0.42567 -0.053 0.958067   
## statela 1.40709 0.39351 3.576 0.000349 \*\*\*  
## stateva -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: 340.04 on 472 degrees of freedom  
## Residual deviance: 275.18 on 469 degrees of freedom  
## AIC: 283.18  
##   
## Number of Fisher Scoring iterations: 6

The summary of the logistic regression model for violator against state shows that the states of Kentucky and Virginia have negative coefficients while Louisana has a positive coefficient. The levels for Louisana and Virgina are very significant while the p-value for Kentucky is quite large. The model above has an AIC value of 283.18. Further analysis and models will need to be completed to determine is an AIC of 283.13 is a strong model or not.

#### Task 4

# Build model with all variables  
allmod = glm(violator ~., train, family = "binomial")   
summary(allmod)

##   
## Call:  
## glm(formula = violator ~ ., family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6055 -0.3932 -0.2643 -0.1384 2.9470   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.750397 1.318165 -2.845 0.00444 \*\*   
## malefemale 0.137577 0.411340 0.334 0.73803   
## racenot.white 1.143719 0.403890 2.832 0.00463 \*\*   
## age 0.005279 0.016910 0.312 0.75490   
## stateky 0.124282 0.492370 0.252 0.80072   
## statela 0.217202 0.556154 0.391 0.69614   
## stateva -3.801561 0.666733 -5.702 1.19e-08 \*\*\*  
## time.served -0.109344 0.118901 -0.920 0.35777   
## max.sentence 0.065956 0.054593 1.208 0.22700   
## multiple.offensesyes 1.711032 0.396463 4.316 1.59e-05 \*\*\*  
## crimedrug 0.516479 0.739095 0.699 0.48468   
## crimeother 0.727043 0.690775 1.053 0.29257   
## crimelarceny 1.119953 0.797552 1.404 0.16025   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 340.04 on 472 degrees of freedom  
## Residual deviance: 242.09 on 460 degrees of freedom  
## AIC: 268.09  
##   
## Number of Fisher Scoring iterations: 6

The summary of a logistic regression model using all variables produces an AIC of 268.09 which is an improvement on the first model above using only state as a predictor. Examaning the model coefficients, we can see that multiple offenses appears to be significant as well as race. Interestingly, the level for Virginia appears to be significant. We will perform a backwards stepwise feature selection to determine the model combination of variables for the model.

# backward stepwise approach  
backmod = stepAIC(allmod, direction = "backward", trace = TRUE)

## Start: AIC=268.09  
## violator ~ male + race + age + state + time.served + max.sentence +   
## multiple.offenses + crime  
##   
## Df Deviance AIC  
## - crime 3 244.47 264.47  
## - age 1 242.18 266.18  
## - male 1 242.20 266.20  
## - time.served 1 242.93 266.93  
## - max.sentence 1 243.57 267.57  
## <none> 242.09 268.09  
## - race 1 250.24 274.24  
## - multiple.offenses 1 261.96 285.96  
## - state 3 316.24 336.24  
##   
## Step: AIC=264.47  
## violator ~ male + race + age + state + time.served + max.sentence +   
## multiple.offenses  
##   
## Df Deviance AIC  
## - age 1 244.48 262.48  
## - male 1 244.85 262.85  
## - time.served 1 245.04 263.04  
## - max.sentence 1 246.00 264.00  
## <none> 244.47 264.47  
## - race 1 252.62 270.62  
## - multiple.offenses 1 265.46 283.46  
## - state 3 321.69 335.69  
##   
## Step: AIC=262.48  
## violator ~ male + race + state + time.served + max.sentence +   
## multiple.offenses  
##   
## Df Deviance AIC  
## - male 1 244.86 260.86  
## - time.served 1 245.04 261.04  
## - max.sentence 1 246.01 262.01  
## <none> 244.48 262.48  
## - race 1 252.65 268.65  
## - multiple.offenses 1 265.52 281.52  
## - state 3 322.14 334.14  
##   
## Step: AIC=260.86  
## violator ~ race + state + time.served + max.sentence + multiple.offenses  
##   
## Df Deviance AIC  
## - time.served 1 245.31 259.31  
## - max.sentence 1 246.33 260.33  
## <none> 244.86 260.86  
## - race 1 252.80 266.80  
## - multiple.offenses 1 265.93 279.93  
## - state 3 322.54 332.54  
##   
## Step: AIC=259.31  
## violator ~ race + state + max.sentence + multiple.offenses  
##   
## Df Deviance AIC  
## - max.sentence 1 246.98 258.98  
## <none> 245.31 259.31  
## - race 1 253.11 265.11  
## - multiple.offenses 1 266.89 278.89  
## - state 3 323.88 331.88  
##   
## Step: AIC=258.98  
## violator ~ race + state + multiple.offenses  
##   
## Df Deviance AIC  
## <none> 246.98 258.98  
## - race 1 254.96 264.96  
## - multiple.offenses 1 267.66 277.66  
## - state 3 332.93 338.93

summary(backmod)

##   
## Call:  
## glm(formula = violator ~ race + state + multiple.offenses, family = "binomial",   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.3609 -0.4094 -0.2705 -0.1575 2.9653   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.51087 0.36354 -6.907 4.96e-12 \*\*\*  
## racenot.white 1.09382 0.38974 2.807 0.00501 \*\*   
## stateky 0.07372 0.46051 0.160 0.87282   
## statela 0.10381 0.50018 0.208 0.83559   
## stateva -3.60795 0.63788 -5.656 1.55e-08 \*\*\*  
## multiple.offensesyes 1.73482 0.39421 4.401 1.08e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 340.04 on 472 degrees of freedom  
## Residual deviance: 246.98 on 467 degrees of freedom  
## AIC: 258.98  
##   
## Number of Fisher Scoring iterations: 6

The backwards stepwise approach selected a combination of variables that produced the best AIC score. We can see that the model now has an AIC value of 258.98 which is ten points lowers than the full model. This model uses race, state and multiple offenses to predict whether or not a person will violate the terms of their parole. The model coefficients are supported by the graphs above. Multiple offenses appeared to be significant in the graph and now the summary confirms this. The same can be applied to state and race.

#### Task 5

# Create model from backwards stepwise  
mod2 <- glm(violator ~ race + state + multiple.offenses,  
 train, family = "binomial")   
summary(mod2)

##   
## Call:  
## glm(formula = violator ~ race + state + multiple.offenses, family = "binomial",   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.3609 -0.4094 -0.2705 -0.1575 2.9653   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.51087 0.36354 -6.907 4.96e-12 \*\*\*  
## racenot.white 1.09382 0.38974 2.807 0.00501 \*\*   
## stateky 0.07372 0.46051 0.160 0.87282   
## statela 0.10381 0.50018 0.208 0.83559   
## stateva -3.60795 0.63788 -5.656 1.55e-08 \*\*\*  
## multiple.offensesyes 1.73482 0.39421 4.401 1.08e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 340.04 on 472 degrees of freedom  
## Residual deviance: 246.98 on 467 degrees of freedom  
## AIC: 258.98  
##   
## Number of Fisher Scoring iterations: 6

This is the same model that was produced using the backwards stepwise approach.

#### Task 6

# Make first prediction  
newdata = data.frame(race = "white", state = "la", multiple.offenses = "yes")  
predict(mod2, newdata, type = "response")

## 1   
## 0.3379961

According to the model, a person from Louisana who is white and has committed multiple offenses has a 33.8 percent chance of violating their parole. Let’s make a second prediction.

newdata1 = data.frame(race = "not.white", state = "ky", multiple.offenses = "no")  
predict(mod2, newdata1, type = "response")

## 1   
## 0.2069629

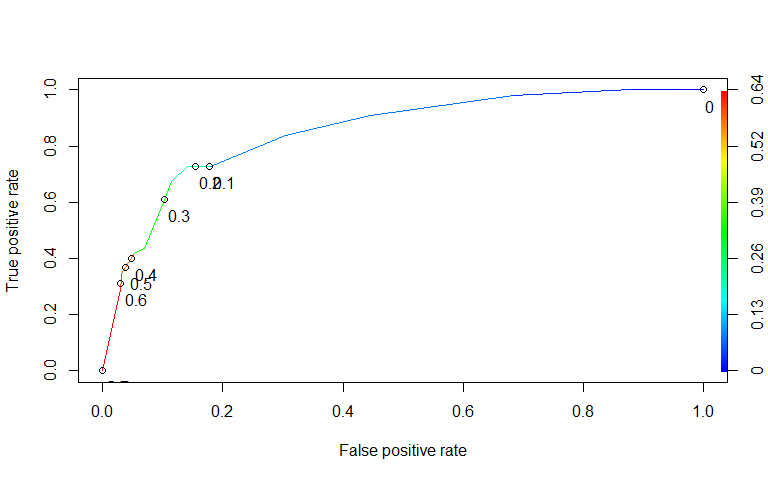
The second prediction is a non-white person from Kentucky who has not committed multiple offenses and they have a 20.7 percent chance of violating parole, according to the model.

#### Task 7

# Develop predictions on training data set  
predictions = predict(mod2, type="response")  
head(predictions)

## 1 2 3 4 5 6   
## 0.07509978 0.19512504 0.19512504 0.07509978 0.07509978 0.19512504

# Create ROCR object  
ROCRpred = prediction(predictions, train$violator)   
  
# Create ROC plot  
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))



Based on the ROC plot above, it appears that the cutoff value should be around 0.2 to best balance sensitivity and specificity.

#### Task 8

# Calculate sensitivity, specificity and cutoff value  
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 0.7272727  
## specificity 0.8588517  
## cutoff 0.2069629

# Create confusion matrix  
t1 = table(train$violator,predictions > 0.2069629)  
t1

##   
## FALSE TRUE  
## no 359 59  
## yes 15 40

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

## [1] 0.8435518

The code above identifies the optimal cutoff value of 0.207 which is incredibly close to the value estimated in Task 7. Using that as a cutoff value, we see a sensitivity of 0.727 meaning that the model correctly identified nearly 73 percent of those who violated their parole. The model has a specificity of 0.859 meaning that it correctly identified 86 percent of those who did not violate their parole. Finally, we can calculate the model accuracy which reveals that it correctly identified 84.4 percent of values. However, we must remember that there are fewer observations of those who did violate parole than those who did not violate parole. We may want to be overly sensitive to ensure the model correctly identifies those who will violate parole. While this would be done to maintain order in society, it may also place unnecessary restrictions or limit the possibilites for those who will not violate parole.

#### Task 9

Using a trial and error approach, we will try to identify a probability threshold that maximizes accuracy alone.

# Using 50 percent as the cutoff  
t2 = table(train$violator,predictions > 0.5)  
t2

##   
## FALSE TRUE  
## no 405 13  
## yes 36 19

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

## [1] 0.8964059

# Using 60 percent as the cutoff  
t3 = table(train$violator,predictions > 0.6)  
t3

##   
## FALSE TRUE  
## no 406 12  
## yes 39 16

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

## [1] 0.8921776

# Using 40 percent as the cutoff  
t4 = table(train$violator,predictions > 0.4)  
t4

##   
## FALSE TRUE  
## no 405 13  
## yes 36 19

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

## [1] 0.8964059

Using 50 percent as the probability threshold, we can see this generates a model with an accuracy of 89.6 percent. If we change that value to 60 percent, the accuracy decreases slightly. If we adjust the threshold to 40 percent, we see the same level of accuracy as the 50 percent threshold.

#### Task 10

# Run model on test data  
test.pred <- predict(mod2, test, type = "response")  
  
# Create confusion matrix  
t5 = table(test$violator,test.pred > 0.5)  
t5

##   
## FALSE TRUE  
## no 174 5  
## yes 15 8

# Calculate accuracy  
(t5[1,1]+t5[2,2])/nrow(test)

## [1] 0.9009901

Running the model on the test data and using the probability threshold that maximizes accuracy in the train data set, we can see that it scores 90 percent accuracy on the test data set.