Project B: Criminal Recidivism Prediction

Kaila Gilbert, Brian Rhindress, Matt Samach

May 3, 2019

Introduction

The goal of this project is to construct various models that could predict violent recidivism and recidivism in general in Broward County, FL. This project occurs against the backdrop of recent research conducted by Propublica on algorithmic bias. When Propublic analyzed COMPAS, a common risk assessment used to score individuals in the criminal justice who are likely to recidivate, the organization found traces of racial bias. In particular, African-Americans were more likely to be misclassified as risky. Caucasian individuals were more likely to be misclassified in the opposite direction.

Our group accrued the same dataset utilized by Propublic to create our own models and compare results, within this context.

This report consists of five key sections:

- Introduction
- Data Cleanup and Transformation
- Exploratory Data Analysis
- Model Selection and Self-Comparison
- Comparisons to COMPAS
- Discussion of Results

Because the primary task of this project centers around model design and performance, we put much effort into exploring, tuning and

analyzing a variety of classification methods. This includes additive logistic regressions, discriminant analyses, KNN, and decision-trees. For each method, we included a confusion matrix and ROC object, from which we could extract key performance metrics. We did not enter this project with strong priors about which method could be best, but rather assumed different methods could perform differentially on key performance metrics.

Most metrics found that the strongest predictors of recidivism were age, sex, and prior involvement with the criminal justice system.

When we reached the end of our analysis, we found that simpler models often performed on par with complex models. A pruned forest worked best for crime in general, but a linear discriminant analysis performed admirably on the violent recidivism outcome. When comparing our highest performing models with the metrics observed from COMPAS performance metrics, we witnessed that we were often able to yield similar rates of sensitivity with more specificity. This was extremely marked once we observed how our models impacted individuals of different sex and race.

Before running, make sure following files are in the current directory:

- violent_ROC_plot.RData
- performance.metrics.violent.RData
- df.twoyear.violent.RData

Data Wrangling

Reading In Files

```
wd = getwd()
```

First, we imported the relevant libraries and read in the file containing data on two-year recidivism.

```
# All below from pro-publica
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(prettydoc)
library(ggplot2)
library(fastDummies)
```

```
## Warning: package 'fastDummies' was built under R version 3.5.3
```

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.5.3
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 3.5.3
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
### READ IN compas-scores-two-years.csv ###
raw_data <- read.csv("https://raw.githubusercontent.com/propublica

## Violent - ran prior, saved key objects, and load when needed ##
# raw_data <- read.csv("https://raw.githubusercontent.com/propubli

# Setting seed for consistency
set.seed(123)

### CREATE DATAFRAME AND FILTER AS DESIRED ###
df <- dplyr::select(raw_data, age, c_charge_degree, race,
```

```
                    age_cat, score_text, sex, priors_count,
                    days_b_screening_arrest, decile_score,
                    is_recid, two_year_recid, c_jail_in, c_jail_ou
  dplyr::filter(days_b_screening_arrest <= 30, days_b_screening_ar
         is_recid != -1, c_charge_degree != "O", score_text != 'N/
  dplyr::select(-days_b_screening_arrest)
nrow(df)
```

```
## [1] 6172
```

We duplicated this for the dataframe associated with violent recidivism.

```
vraw_data <- read.csv("https://raw.githubusercontent.com/propublic

library(dplyr)
vdf <- dplyr::select(vraw_data, age, c_charge_degree, race,
                    age_cat, score_text, sex, priors_count,
                    days_b_screening_arrest, decile_score,
                    is_recid, two_year_recid, c_jail_in, c_jail_ou
  filter(days_b_screening_arrest <= 30, days_b_screening_arrest >=
         is_recid != -1, c_charge_degree != "O", score_text != 'N/
  dplyr::select(-days_b_screening_arrest)
nrow(vdf)
```

```
## [1] 4015
```

Dummies and Additional Features

Below, we undertook some basic data wrangling to ensure we had dummies when necessary for the classification tasks. We also ensured column names were in a form recognizable to R. We did this for both the violent and nonviolent datasets.

```
library(fastDummies)
```

```
## Warning: package 'fastDummies' was built under R version 3.5.3
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
dummy_col_names = c("c_charge_degree","race","age_cat","score_text
df_dummies <- fastDummies::dummy_cols(.data=df,select_columns=dumm
df_dummies <-  df_dummies %>% dplyr:: select(-dummy_col_names)

### how to fix a caret ###
names(df_dummies)[which(names(df_dummies)=="race_African-American"
names(df_dummies)[which(names(df_dummies)=="race_Native American")
colnames(df_dummies)
```

```
##  [1] "age"                       "priors_count"
##  [3] "decile_score"              "is_recid"
##  [5] "two_year_recid"            "c_jail_in"
##  [7] "c_jail_out"                "c_charge_degree_F"
##  [9] "c_charge_degree_M"         "race_Other"
## [11] "race_African_American"     "race_Caucasian"
## [13] "race_Hispanic"             "race_Asian"
## [15] "race_Native_American"      "age_cat_Greater than 45"
## [17] "age_cat_25 - 45"           "age_cat_Less than 25"
## [19] "score_text_Low"            "score_text_Medium"
## [21] "score_text_High"           "sex_Male"
```

```
## [23] "sex_Female"
```

```
#Violent
#Now With Violence
vdummy_col_names = c("c_charge_degree","race","age_cat","score_tex
vdf_dummies <- fastDummies::dummy_cols(.data=vdf,select_columns=vc
vdf_dummies <-  vdf_dummies %>% dplyr:: select(-vdummy_col_names)

names(vdf_dummies)[which(names(vdf_dummies)=="race_African-America
names(vdf_dummies)[which(names(vdf_dummies)=="race_Native Americar
colnames(vdf_dummies)
```

```
##  [1] "age"                       "priors_count"
##  [3] "decile_score"              "is_recid"
##  [5] "two_year_recid"            "c_jail_in"
##  [7] "c_jail_out"                "c_charge_degree_F"
##  [9] "c_charge_degree_M"         "race_Other"
## [11] "race_African_American"     "race_Caucasian"
## [13] "race_Hispanic"             "race_Asian"
## [15] "race_Native_American"      "age_cat_Greater than 45"
## [17] "age_cat_25 - 45"           "age_cat_Less than 25"
## [19] "score_text_Low"            "score_text_High"
## [21] "score_text_Medium"         "sex_Male"
## [23] "sex_Female"
```

```
# From propublica
df_dummies$length_of_stay <- as.numeric(as.Date(df$c_jail_out) - a
df_dummies <- df_dummies %>% dplyr::select(-c("c_jail_out", "c_jai
cor(df_dummies$length_of_stay, df_dummies$decile_score)
```

```
## [1] 0.2073297
```

```
#Violent
vdf_dummies$length_of_stay <- as.numeric(as.Date(vdf$c_jail_out) -
vdf_dummies <- vdf_dummies %>% dplyr::select(-c("c_jail_out", "c_j
cor(vdf_dummies$length_of_stay, vdf_dummies$decile_score)
```

```
## [1] 0.2093402
```

```
#Additional Cleanup; Versions without Dummies
df.twoyear <- df %>% dplyr::select(-c("age_cat", "score_text", "de
                                "c_jail_in", "c_jail_out"))
df.twoyear$two_year_recid <- as.factor(df.twoyear$two_year_recid)
df.twoyear$length_of_stay <- as.numeric(as.Date(df$c_jail_out) - a
```

Exploratory Data Analysis

In the sections below, we sought to gain some understanding of the underlying data. In particular, we were interested in relationships across two-year recidivism and the features available to us in the data. We also include Probublica's visualization of the inconsistencies across score because it was helpful as we thought about structure.

```
# From ProPublica
summary(df$age_cat)
```

```
##         25 - 45 Greater than 45    Less than 25
##            3532            1293            1347
```

```
xtabs(~ sex + race, data=df)
```

```
##         race
## sex      African-American Asian Caucasian Hispanic Native Ameri
##   Female              549     2       482       82
##   Male               2626    29      1621      427
```

```
summary(df$sex)
```

```
## Female   Male
```

```
##   1175   4997
```

```
nrow(filter(df, two_year_recid == 1))
```

```
## [1] 2809
```

```
nrow(filter(df, two_year_recid == 1)) / nrow(df) * 100
```

```
## [1] 45.51199
```

```
#Violent
summary(vdf$age_cat)
```

```
##         25 - 45 Greater than 45    Less than 25
##            2299             950             766
```

```
xtabs(~ sex + race, data=vdf)
```

```
##         race
## sex      African-American Asian Caucasian Hispanic Native Ameri
##   Female              393     1       335       61
##   Male               1525    25      1120      294
```

```
summary(vdf$sex)
```

```
## Female   Male
##    839   3176
```

```
nrow(filter(vdf, two_year_recid == 1))
```

```
## [1] 652
```

```
nrow(filter(vdf, two_year_recid == 1)) / nrow(df) * 100
```

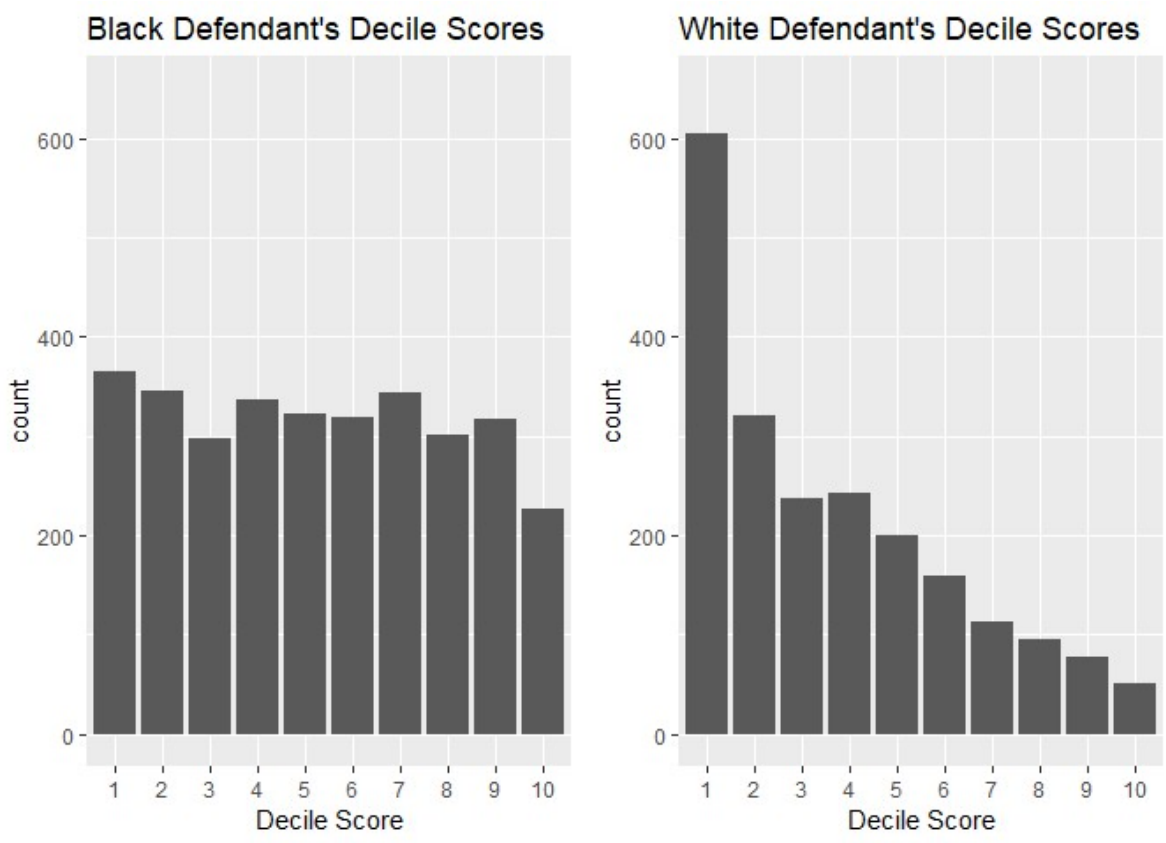
```
## [1] 10.56384
```

```
# From ProPublica
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ggplot2)

pblack <- ggplot(data=filter(df, race =="African-American"), aes(d
          geom_bar() + xlab("Decile Score") +
          ylim(0, 650) + ggtitle("Black Defendant's Decile Scores"
pwhite <- ggplot(data=filter(df, race =="Caucasian"), aes(ordered(
          geom_bar() + xlab("Decile Score") +
          ylim(0, 650) + ggtitle("White Defendant's Decile Scores"
grid.arrange(pblack, pwhite,  ncol = 2)
```



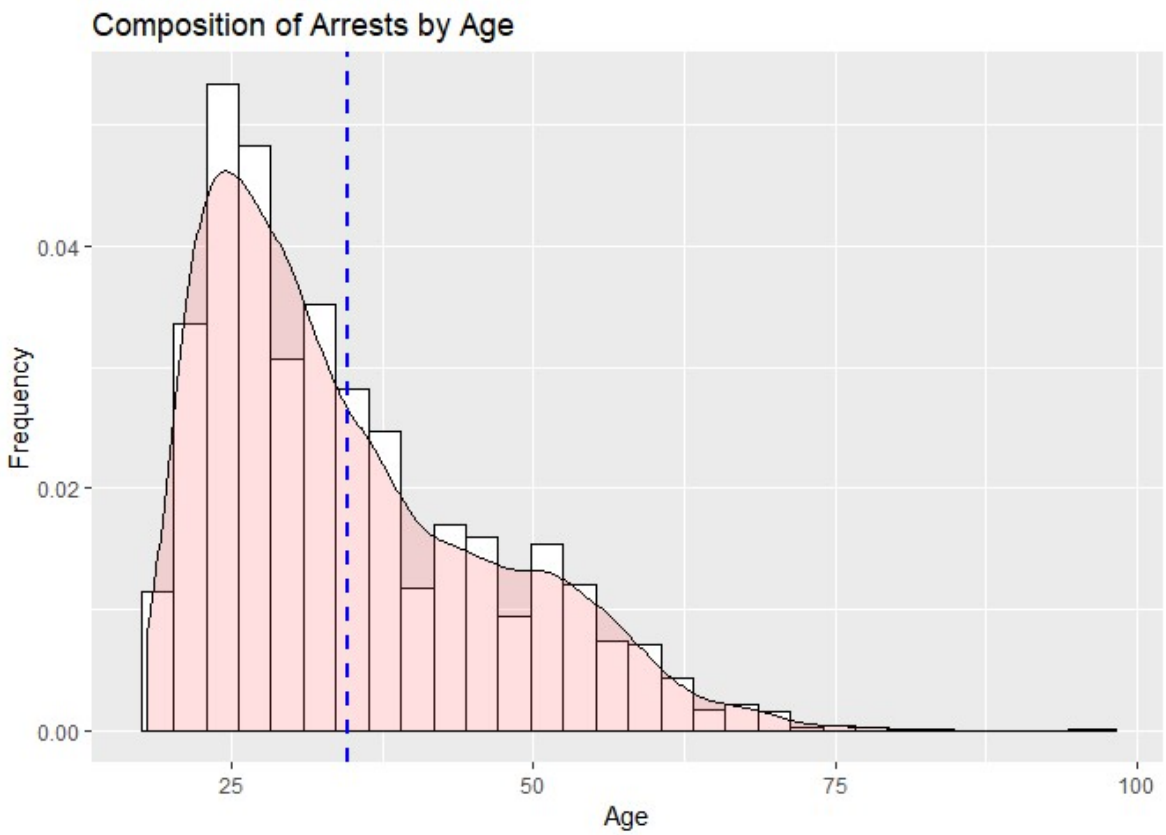
The Propublica analysis showed clear, disparate patterns in the scores of its defendants. Notably, distributions across score appeared nearly uniform for Black Defendants,yet skewed heavily to the right for white defendants. We sought more granular information about the types of individuals who tend to recidivate, and produced a series of visuals and histograms that could communicate the actual, observed two-year recidivism across groups.

In general, we saw that the age of defendants tended to be younger, with a distribution that is strongly skewed to the right. There is a peak around age 25, after which case there is a gradual decline. When overlaying this data with the proportion of each age that recidicated, we observed a similar trend. The recidivism patterns, as a proportion of the age bin, we starkly different. Younger individuals recidivated at nearly even proportions, while this activity fell as age increased.

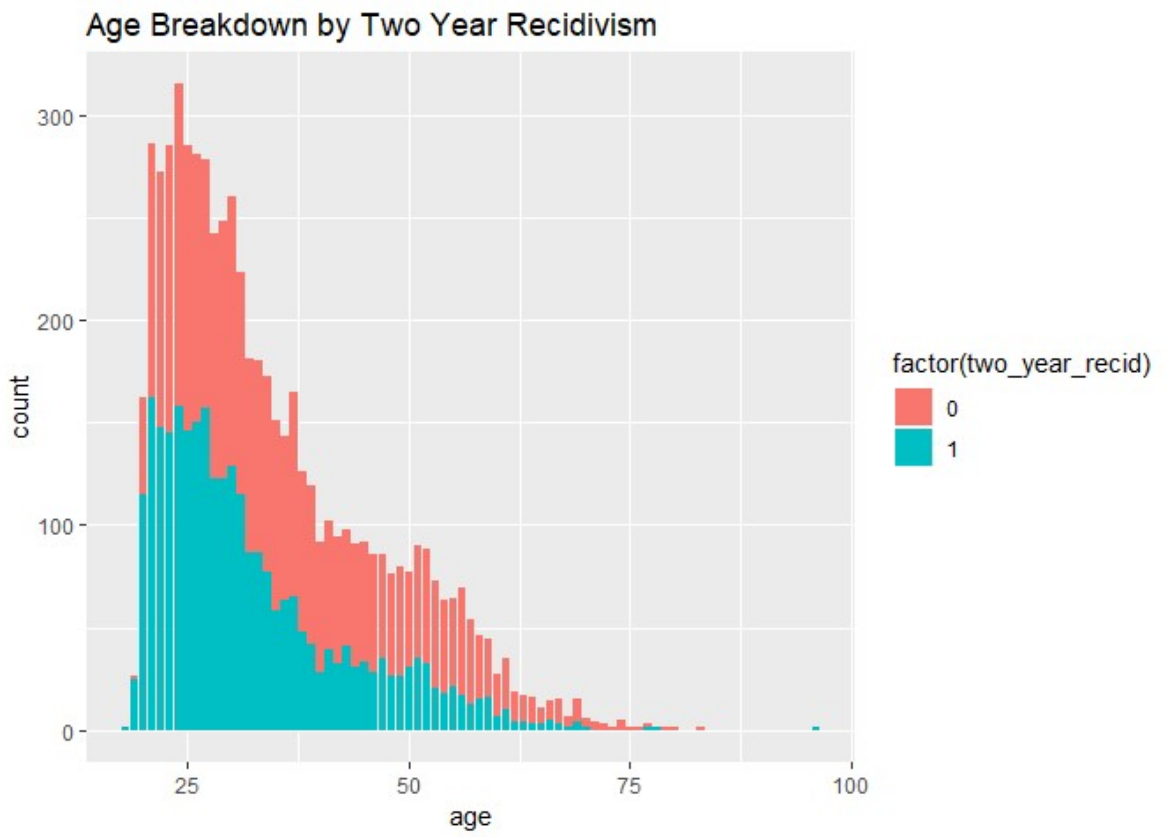
```
library(ggplot2)
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E44
options(scipen = 4)
```

```
#Relationship Between Age, Race, Sex, and Priors and Recidicism

ggplot(df, aes(x=age)) + geom_histogram(aes(y=..density..), colour
 geom_density(alpha=.2, fill="#FF6666") + labs(title="Composition
            color="blue", linetype="dashed", size=1)
```

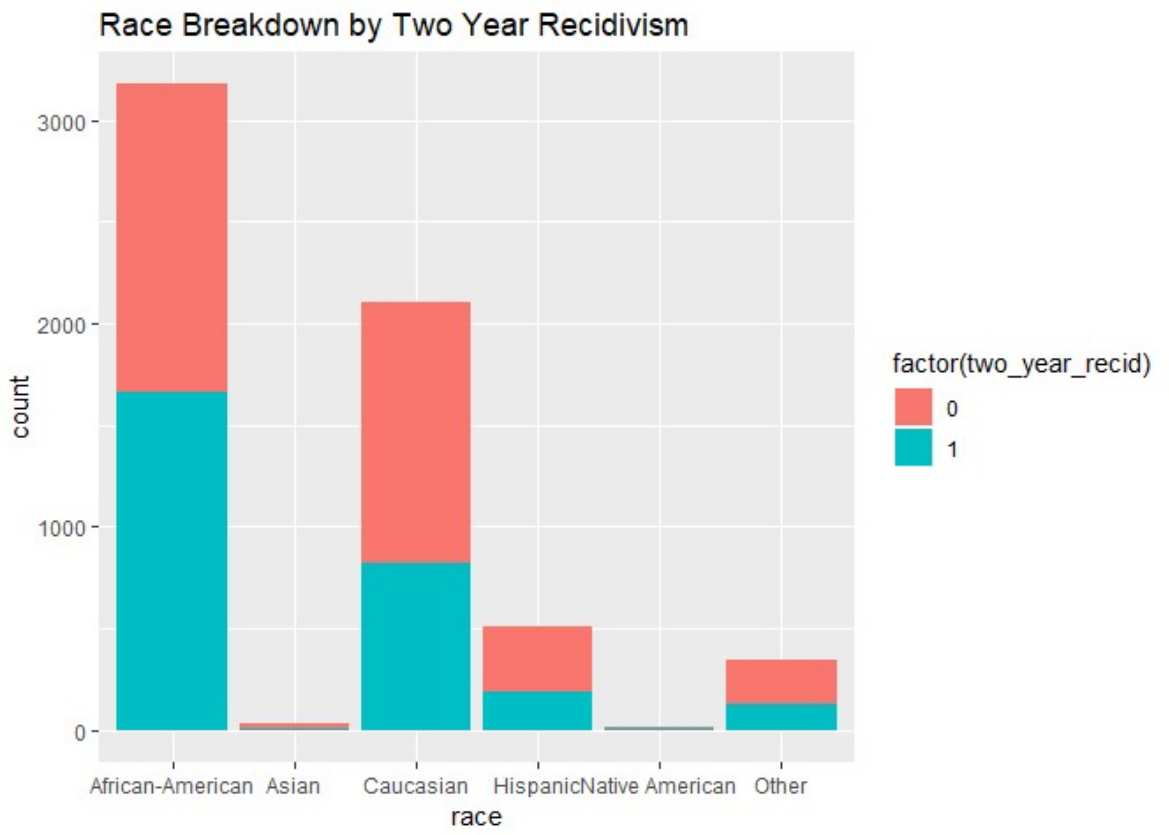
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidt
```



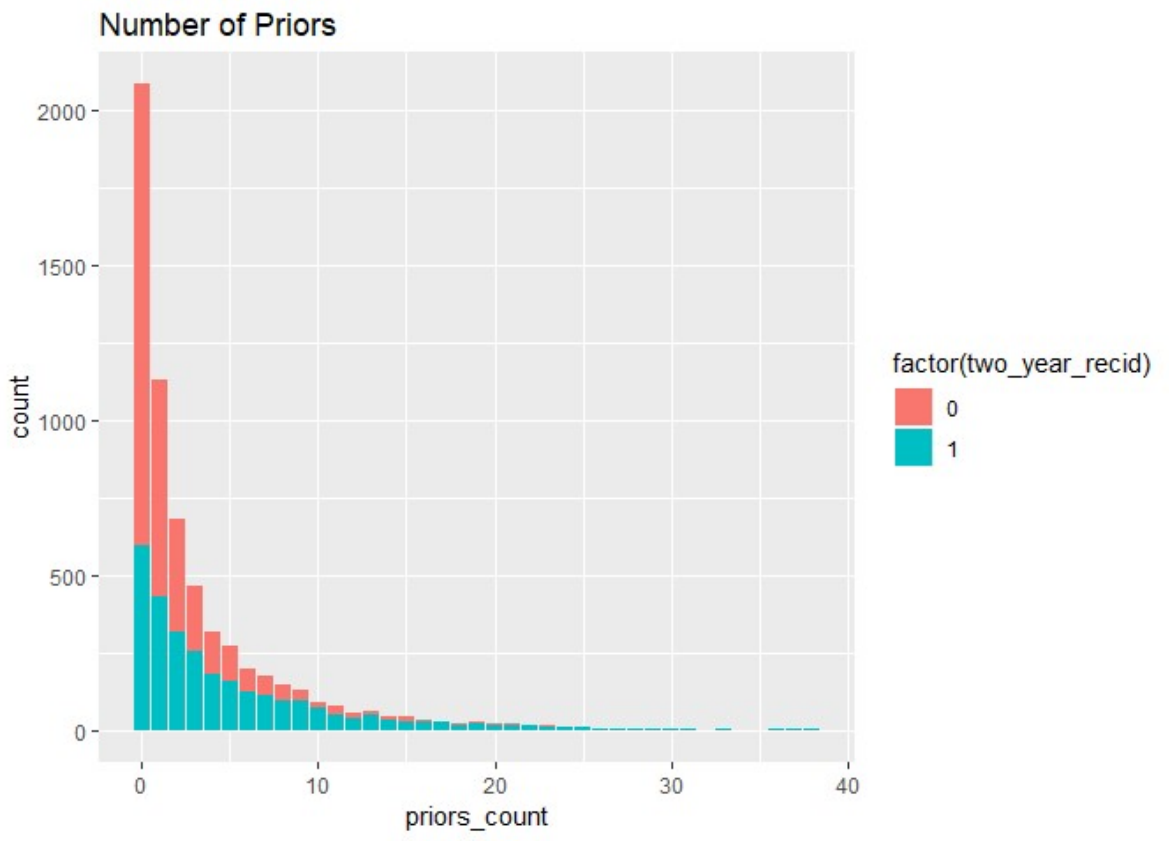

```
ggplot(data=df, mapping=aes(x=age, fill=factor(two_year_recid))) +
```



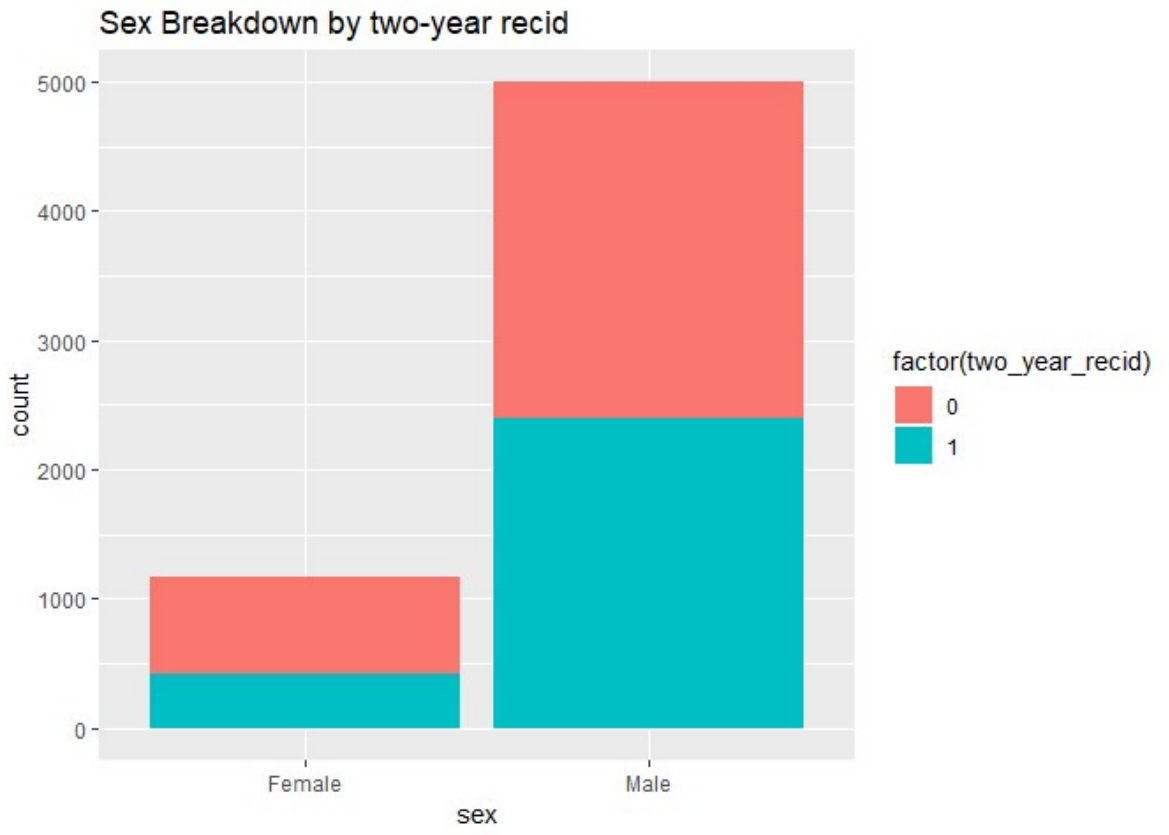

```
ggplot(data=df, mapping=aes(x=race, fill=factor(two_year_recid)))
```




```
ggplot(data=df, mapping=aes(x=priors_count, fill=factor(two_year_r
```




```
ggplot(data=df, mapping=aes(x=sex, fill= factor(two_year_recid)))
```



We also observed breakdowns by race and by sex. The majority of inmates were male, compared to female. They experienced similar recidivism rates as a proportion to their size. For our graph exploring race, we observed that African-Americans were highly represented in the system, exhibiting a plurality of defendants. The second most populous group was Caucasian, with smaller representation of Hispanics, Asians, and those classified as Other.

Because we assumed the past could be some indicator of the present, we also mapped out the number of priors. We observed that the majority of individuals were “first-timers”; the bulk of our dataset had fewer than 2 previous encounters with the criminal justice system. However, as the number of priors increased, naturally, the percentage that recidivated within two years also increased. Therefore, we received some indication that frequent criminal activity is positively correlated with the likelihood to recidivate within two years.

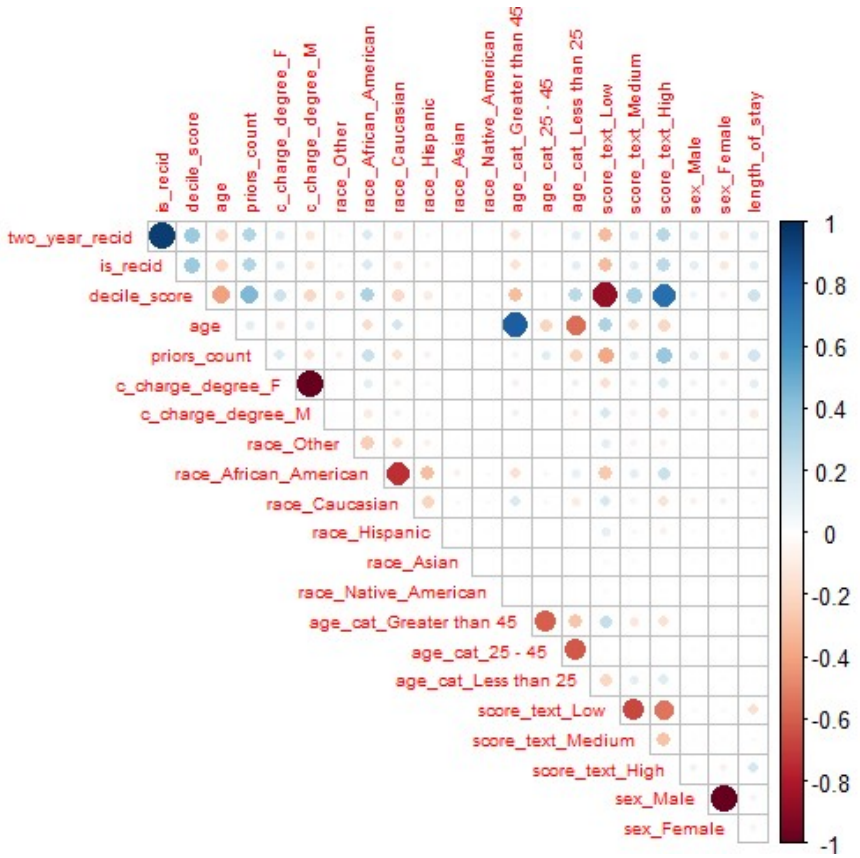
Correlations

We were interested in how these features interacted with each other, so we created correlation plots with both the violent and general recidivism data. The plots are shown below.

```
library(corrplot)
```

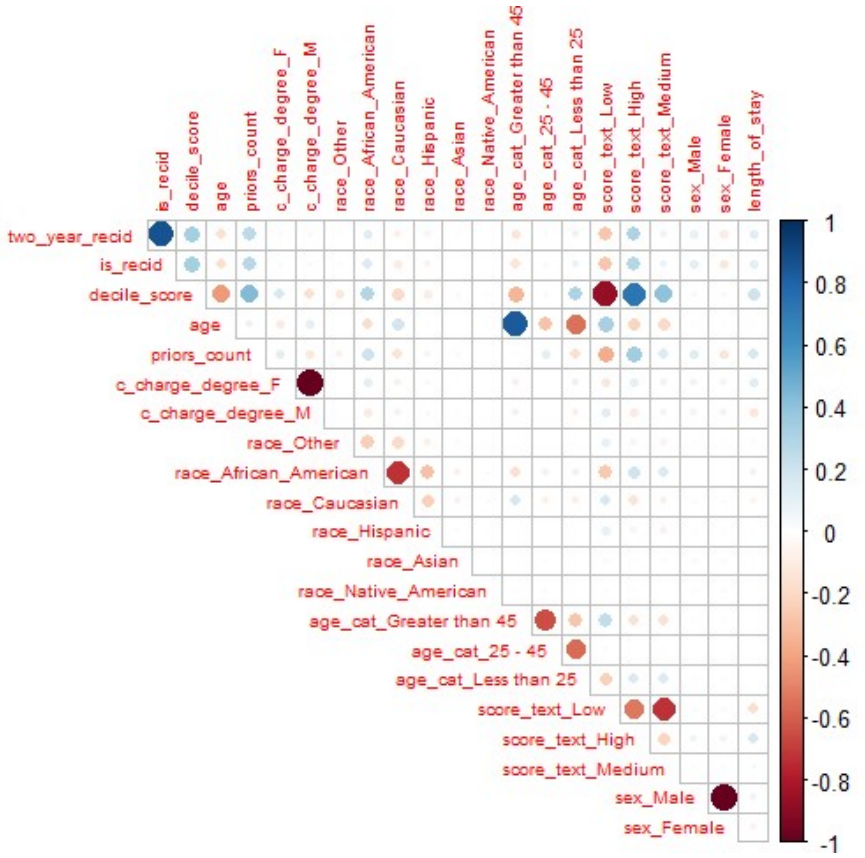
```
## corrplot 0.84 loaded
```

```
# reorder so outputs of interest are at the beginning
df_dummies <- df_dummies %>% dplyr::select("two_year_recid", "is_r
                                           everything())
correlations <- cor(df_dummies, use = "pairwise.complete.obs")

corrplot(correlations, diag = FALSE, type = "upper", tl.cex = 0.6)
```




```
#For Violent
vdf_dummies <- vdf_dummies %>% dplyr::select("two_year_recid", "is
```

```
vcorrelations <- cor(vdf_dummies, use = "pairwise.complete.obs")
corrplot(vcorrelations, diag = FALSE, type = "upper", tl.cex = 0.6
```



The first thing we noticed is that several of the original predictors were almost perfectly collinear or non-collinear. In particular, indicators related to age, score, sex, and race were all strongly correlated with one another. Because this indicated that these variables were all essentially capturing the same phenomenon, we used this guidance to eliminate redundant variables.

Beyond perfectly collinear variables, a few other trends emerged. In particular, there were strong, negative correlations between African-American defendants and a likelihood of a low score. Scores themselves were correlated with recidivism in ways we might expect, but in moderate ways. This correlation plot revealed that low scores were negatively correlated with recidivism and high scores were positively correlated with recidivism. This is important, because it suggests the COMPAS scores are able to pick up some key, meaningful behavioral differences in individuals who are likely (or not) to recidivate. However,

the magnitude of this correlation was not particular strong, and we were curious about how the score seemed to weight the features in our model.

Variable Selection

Before running models, we wanted to observe what variables were most important. We ran exhaustive tests using regsubsets on both datasets. We were also curious what factors predicted socring mechanisms, so we included those in the results.

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.5.3
```

```
# assuming we can't know is_recid as a predictor at time of arrest
# must get rid of highly collinear vars, non predictive vars, othe
twoyear.predictors <- df_dummies %>% dplyr::select(-c("is_recid",
                                                      "score_text_Low", "
                                                      "score_text_High",
                                                      "race_Caucasian", "
                                                      "age_cat_25 - 45",
                                                      "sex_Female"))
twoyear.predictors$two_year_recid <- as.factor(twoyear.predictors$


score.predictors <- df_dummies %>% dplyr::select(-c("is_recid", "t
                                                      "score_text_Low", "
                                                      "score_text_High",
                                                      "race_Caucasian", "
                                                      "age_cat_25 - 45",
                                                      "sex_Female"))
violent_recid.predictors <- vdf_dummies %>% dplyr::select(-c("is_r
                                                      "score_text_Low", "
                                                      "score_text_High",
                                                      "race_Caucasian", "
```

```
                                                "age_cat_25 - 45",
                                                "sex_Female"))
# Getting variables for two year recid
subset.twoyear <- regsubsets(two_year_recid ~ .,
                             data = twoyear.predictors,
                             nbest = 1,
                             nvmax = 12,
                             method = "exhaustive")


# Getting variables for score decile
subset.score <- regsubsets(decile_score ~ .,
                           data = score.predictors,
                           nbest = 1,
                           nvmax = 12,
                           method = "exhaustive")
# Getting variables for violent recid
subset.violent_recid <- regsubsets(two_year_recid ~ .,
                             data = violent_recid.predictors,
                             nbest = 1,
                             nvmax = 12,
                             method = "exhaustive")
```

Upon receiving the top predictors for each model size, we created a dataframe of all results.

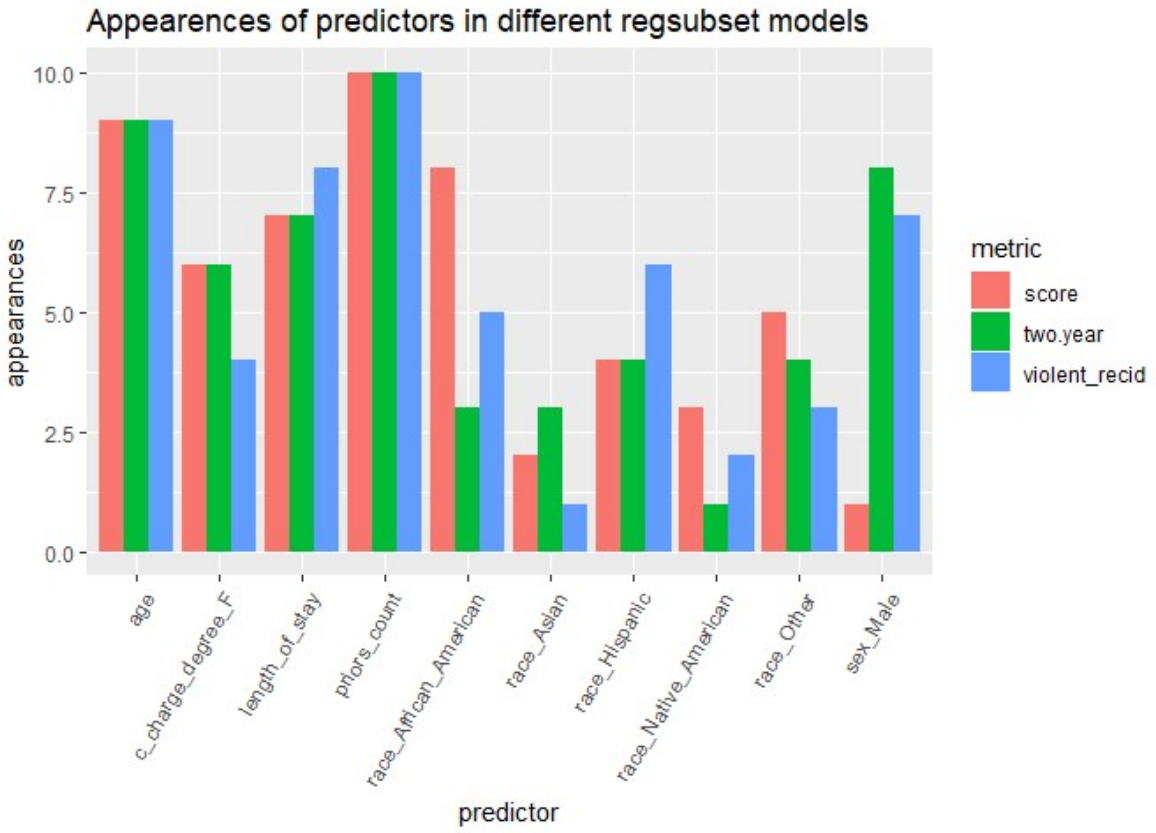
```
twoyear.sum <- summary(subset.twoyear)
predictors <- colnames(twoyear.sum$which[,2:length(colnames(twoyea

twoyear.appearances <- sapply(as.data.frame(twoyear.sum$which), su
twoyear.appearances <- twoyear.appearances[2:length(twoyear.appear

score.sum <- summary(subset.score)
score.appearances <- sapply(as.data.frame(score.sum$which), sum)
score.appearances <- score.appearances[2:length(score.appearances)

violent_recid.sum <- summary(subset.violent_recid)
violent_recid.appearances <- sapply(as.data.frame(violent_recid.su
violent_recid.appearances <- violent_recid.appearances[2:length(vi
```

```
appearance.df <- data.frame("predictor" = names(score.appearances)
                            "appearances" = score.appearances,
                            "metric" = rep("score", length(names(s

appearance.df <- rbind(appearance.df,
                       data.frame("predictor" = names(twoyear.appe
                                  "appearances" = twoyear.appearar
                                  "metric" = rep("two.year",
                                                 length(names(twoy

appearance.df <- rbind(appearance.df,
                       data.frame("predictor" = names(violent_reci
                                  "appearances" = violent_recid.ap
                                  "metric" = rep("violent_recid",
                                                 length(names(viol

rownames(appearance.df) <- c()
```

Regsubsets Analysis

Upon running exhaustive regsubsets models on both datasets, we observed that the subset feature selected age, priors, and sex as their strongest predictors. When overlayed this data with the top features observed in the COMPAS models, we received strongly conflicting information concerning weights.

```
ggplot(data = appearance.df) +
  geom_bar(mapping = aes(x = predictor, y = appearances, fill = me
           position = "dodge", stat = "identity")+
  theme(axis.text.x = element_text(angle = 60, hjust = 1))+
  ggtitle("Appearences of predictors in different regsubset models
```



Model Comparison

Below, we created a common training and test set using 80/20% of our data, respectively, which could be used across a slew of modeling options. Importantly, these training and test sets were randomized and identical across all initial model tests.

```
smp_size <- floor(0.80 * nrow(df))
train_ind <- sample(1:(nrow(df)), size = smp_size)

train <- df.twoyear[train_ind, ]
test <- df.twoyear[-train_ind, ]

x_train <- train[,-6]
y_train <- train$two_year_recid
x_test <- train[,-6]
y_test <- test$two_year_recid

#---------------Unique Test and Train Set for Level-Requiring form
```

```
pract <- df.twoyear
levels <- unique(pract$two_year_recid)
pract$two_year_recid <- factor(pract$two_year_recid, labels=make.r

ctrain <- pract[train_ind, ]
ctest <- pract[-train_ind, ]

cx_train <- ctrain[,-6]
cy_train <- ctrain$two_year_recid
cx_test <- ctest[,-6]
cy_test <- ctest$two_year_recid
```

Discriminant Analysis

First, we ran a linear discriminant analysis and quadratic discriminant analysis without including tuning parameters.

```
library(klaR)
library(caret)
```

```
## Loading required package: lattice
```

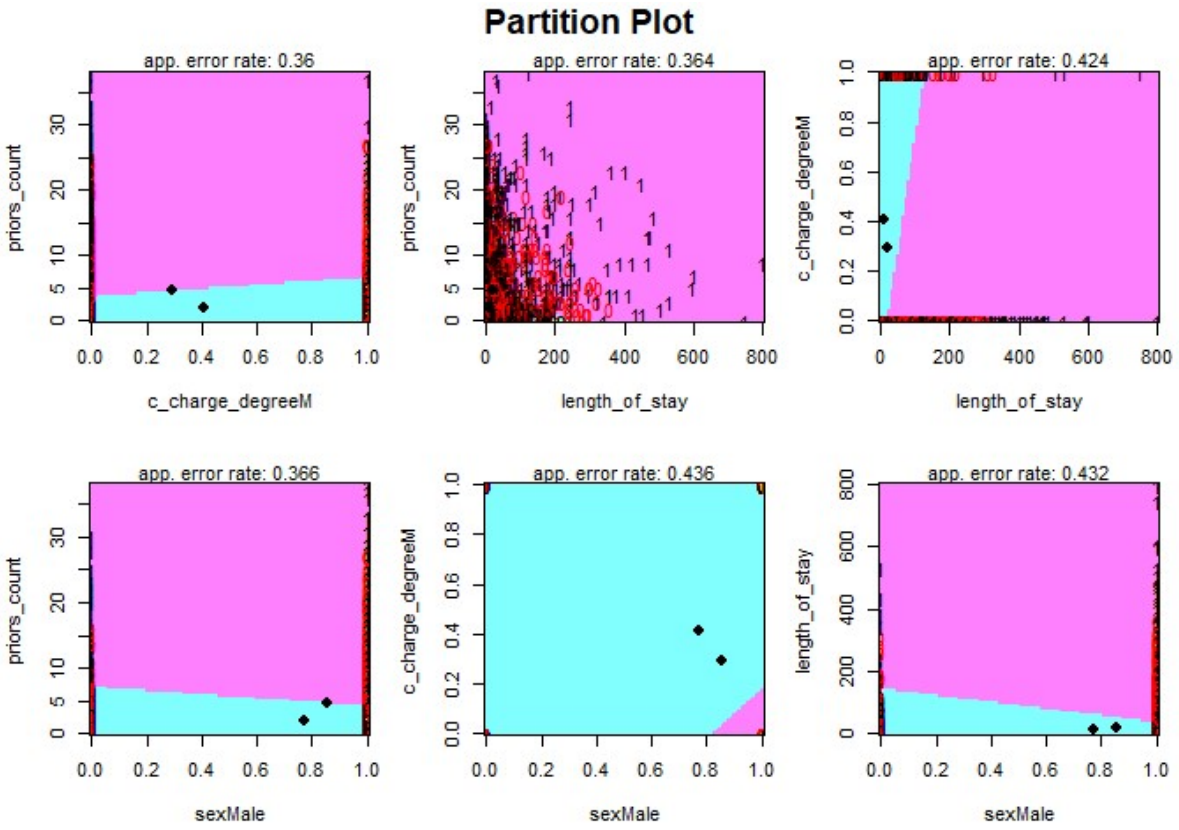
```
#Traditional - Default Settings w/ No tuning parameters
  lda_fit <- lda(two_year_recid ~., data=train)
  qda_fit <- qda(two_year_recid ~., data=train)
  lda_fit$prior #Prevalence
```

```
##         0         1
## 0.5428398 0.4571602
```

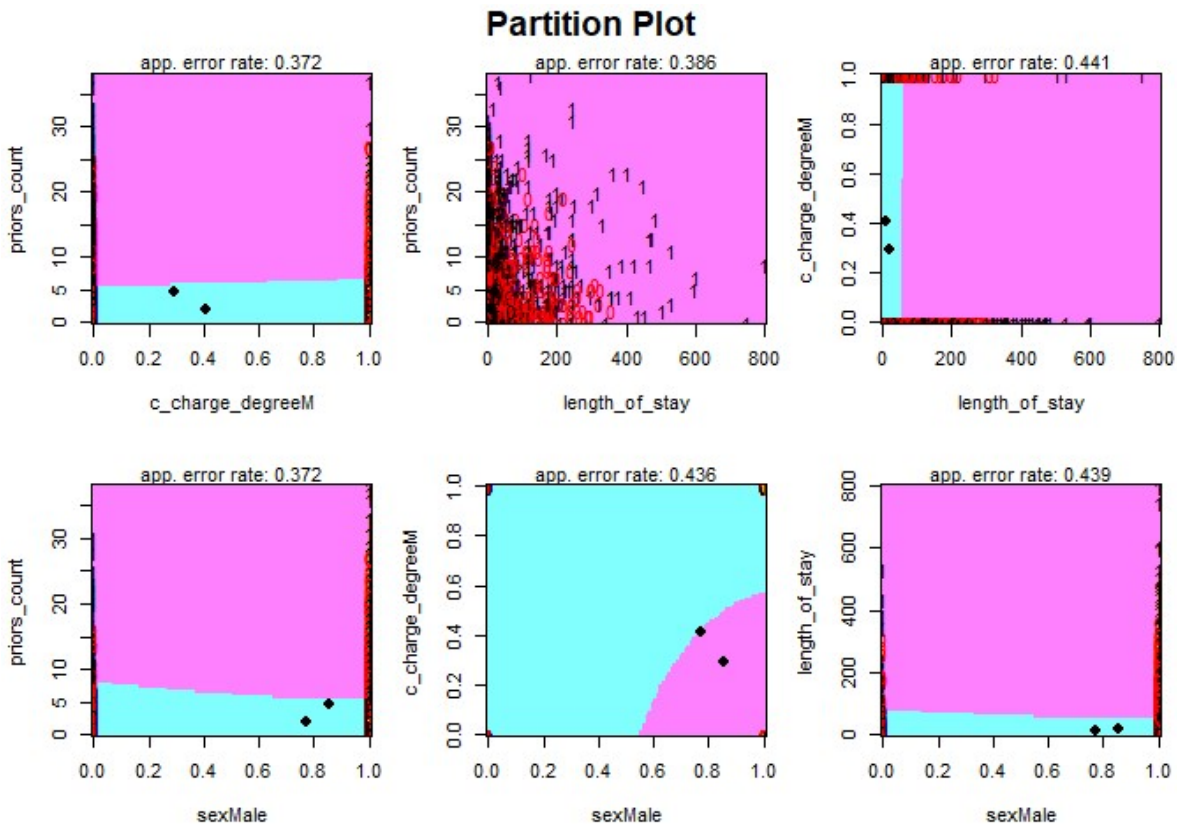
```
  lda_predict <- predict(lda_fit, test, type="response")
  qda_predict <- predict(qda_fit, test)
  qda_pred <- as.numeric(qda_predict$class)
  lda_fit$means
```

```
##        age c_charge_degreeM   raceAsian raceCaucasian raceHispa
## 0 36.56455        0.4041045 0.006716418     0.3798507   0.09440
## 1 32.07709        0.2924236 0.003544528     0.2888790   0.06778
##   raceNative American  raceOther   sexMale priors_count length_
## 0         0.001865672 0.06716418 0.7798507     1.997015       1
## 1         0.001772264 0.04607887 0.8577758     4.711121       2
```

```
#LDA / QDA Plots with Best Variables from Lasso
#Images Show Binary classifications make it unclear where distinct
lda_plots <- partimat(two_year_recid ~ priors_count + c_charge_deg
```




```
qda_plots <- partimat(two_year_recid ~ priors_count + c_charge_deg
```




```
lda_cm <- caret::confusionMatrix(positive = "1", lda_predict$class
qda_cm <- caret::confusionMatrix(positive = "1", qda_predict$class

bin <- as.integer(y_test)
lda_bin <- as.integer(lda_predict$class)
qda_bin <- as.integer(qda_predict$class)

#lda_roc(lda_fit)
lda_roc <-roc(bin, lda_predict$posterior[,2])
qda_roc <- roc(bin, qda_predict$posterior[,2])
```

We found it useful to visualize what the best of a rather simple model could do. When we mapped out the decision boundaries produced by either method, we immediately realized that the method of classification might not be the best, given the diffuseness of individuals on either side of the decision boundary.

Below, we tuned the LDA function with built-in feature selection tools.

```
#Stepwsie LDA with own feature selection
train_control = trainControl(method="CV", number=10, classProbs=TR

lda_stepwise <- train(cx_train,cy_train,
                              method = "stepLDA",
                              trControl = train_control,
                              metric = "ROC")
```

```
##  `stepwise classification', using 10-fold cross-validated corre
```

```
## 4443 observations of 6 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## correctness rate: 0.63359;  in: "priors_count";  variables (1):
##
##  hr.elapsed min.elapsed sec.elapsed
##        0.00        0.00        2.27
```

```
##  `stepwise classification', using 10-fold cross-validated corre
```

```
## 4443 observations of 6 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## correctness rate: 0.63111;  in: "priors_count";  variables (1):
##
##  hr.elapsed min.elapsed sec.elapsed
##         0.0         0.0         1.6
```

```
##  `stepwise classification', using 10-fold cross-validated corre
```

```
## 4444 observations of 6 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## correctness rate: 0.63501;  in: "priors_count";  variables (1):
##
##  hr.elapsed min.elapsed sec.elapsed
##        0.00        0.00        2.12
```

```
##  `stepwise classification', using 10-fold cross-validated corre
```

```
## 4443 observations of 6 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## correctness rate: 0.63178;  in: "priors_count";  variables (1):
##
##  hr.elapsed min.elapsed sec.elapsed
##         0.0         0.0         1.5
```

```
##  `stepwise classification', using 10-fold cross-validated corre
```

```
## 4443 observations of 6 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## correctness rate: 0.63584;  in: "priors_count";  variables (1):
##
##  hr.elapsed min.elapsed sec.elapsed
##        0.00        0.00        1.64
```

```
##  `stepwise classification', using 10-fold cross-validated corre
```

```
## 4443 observations of 6 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## correctness rate: 0.62683;  in: "priors_count";  variables (1):
##
##  hr.elapsed min.elapsed sec.elapsed
##        0.00        0.00        1.66
```

```
##  `stepwise classification', using 10-fold cross-validated corre
```

```
## 4444 observations of 6 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## correctness rate: 0.63255;  in: "priors_count";  variables (1):
##
##  hr.elapsed min.elapsed sec.elapsed
##        0.00        0.00        1.67
```

```
##  `stepwise classification', using 10-fold cross-validated corre
```

```
## 4444 observations of 6 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## correctness rate: 0.62781;  in: "priors_count";  variables (1):
##
```

```
##  hr.elapsed min.elapsed sec.elapsed
##        0.00        0.00        1.48
```

```
##  `stepwise classification', using 10-fold cross-validated corre
```

```
## 4443 observations of 6 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## correctness rate: 0.63155;  in: "priors_count";  variables (1):
##
##  hr.elapsed min.elapsed sec.elapsed
##        0.00        0.00        1.36
```

```
##  `stepwise classification', using 10-fold cross-validated corre
```

```
## 4443 observations of 6 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## correctness rate: 0.62953;  in: "priors_count";  variables (1):
##
##  hr.elapsed min.elapsed sec.elapsed
##        0.00        0.00        1.38
```

```
##  `stepwise classification', using 10-fold cross-validated corre
```

```
## 4937 observations of 6 variables in 2 classes; direction: both
```

```
## stop criterion: improvement less than 5%.
```

```
## correctness rate: 0.63156;  in: "priors_count";  variables (1):
##
##  hr.elapsed min.elapsed sec.elapsed
##         0.0         0.0         1.3
```

```
best_lda <- lda_stepwise[["finalModel"]][["formula"]]
print(best_lda)
```

```
## y ~ priors_count
## <environment: 0x0000000005e2e2c0>
```

```
best_lda <-lda(two_year_recid ~priors_count, data=ctest)
best_lda_pred <- predict(best_lda, ctest, type="raw")
print(best_lda)
```

```
## Call:
## lda(two_year_recid ~ priors_count, data = ctest)
##
## Prior probabilities of groups:
##        X0        X1
## 0.5530364 0.4469636
##
## Group means:
##    priors_count
## X0     1.945827
## X1     4.932971
##
## Coefficients of linear discriminants:
##                     LD1
## priors_count 0.2201716
```

```
#Confusion Matrix
pred = best_lda_pred[["class"]]
pred = plyr::revalue(pred, c("X0"="0", "X1"="1"))
temp_test = plyr::revalue(cy_test, c("X0"="0", "X1"="1"))
```

```
lda_stepwise_cm <- caret::confusionMatrix(pred, temp_test,positive

#ROCstep_bin, test_bin
lda_stepwise_roc <-roc(cy_test, best_lda_pred$posterior[,2])
print(lda_stepwise_roc)
```

```
##
## Call:
## roc.default(response = cy_test, predictor = best_lda_pred$poste
##
## Data: best_lda_pred$posterior[, 2] in 683 controls (cy_test X0)
## Area under the curve: 0.6884
```

Cite: https://github.com/topepo/caret/blob/master/RegressionTests/Code/stepLDA.R

Naive Bayes

For good measure, and for simplicity, we also ran a Naive Bayes test on the dataset. Ultimately, the three methods performed OK.

```
library(caret)
library(klaR)
#Naive Bayes
    #Traditional
nb_fit <- NaiveBayes(two_year_recid ~ priors_count + c_charge_degr
nb_predict <- predict(nb_fit, test, type="response")
nb_cm <- confusionMatrix(positive = "1", nb_predict$class, test$tw

    #Tuning
nb_pred <- rep(0, nrow(test))
nb_pred[nb_predict$posterior[,2] > 0.5] = 1

nb_roc <- roc(test$two_year_recid, nb_predict$posterior[,1])
```

```
library(pROC)
```

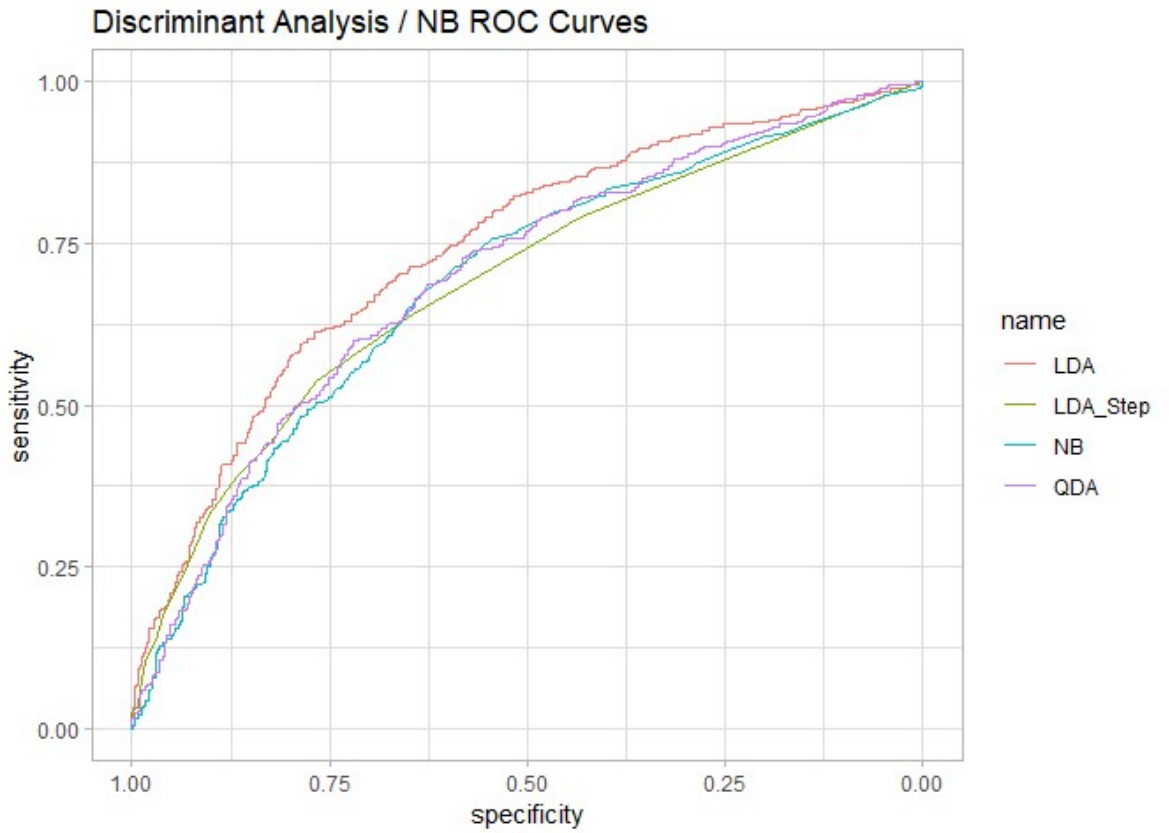
```
## Warning: package 'pROC' was built under R version 3.5.3
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
ggroc(list(NB = nb_roc, LDA = lda_roc, QDA=qda_roc, LDA_Step = lda
```



LDA appeared to be the best classifier among the different permutations of these groups. This also held true in the violent recidivism dataset.

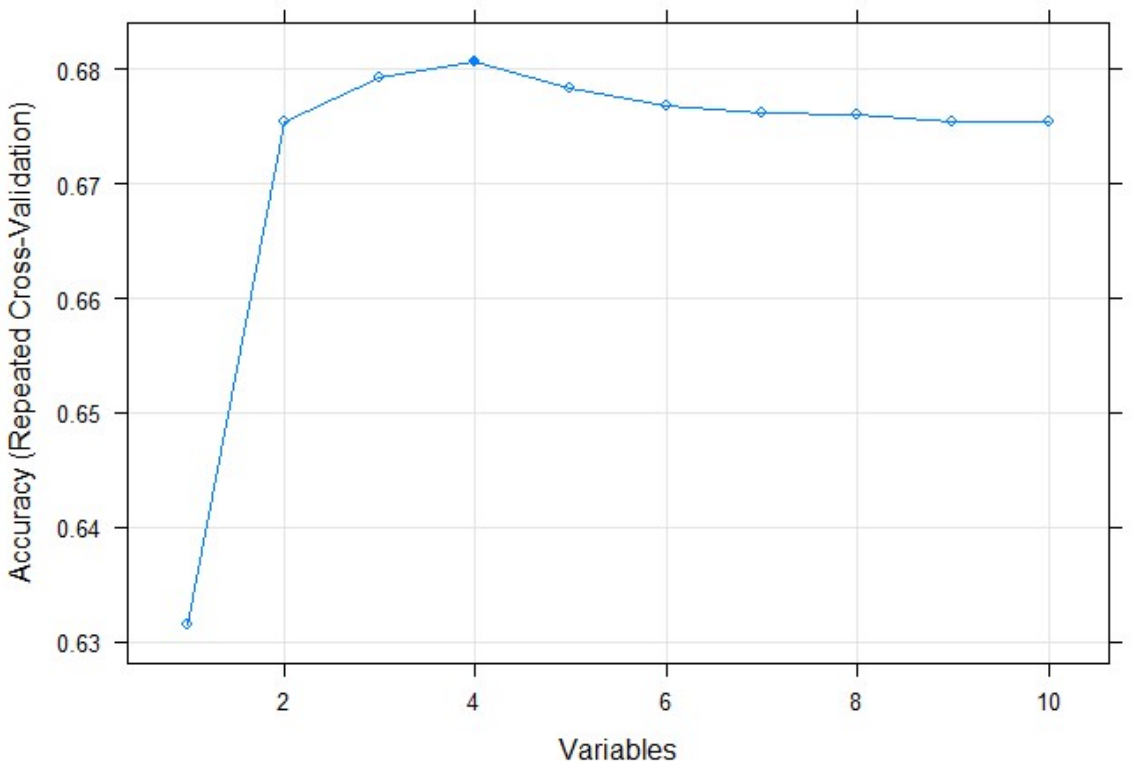
We next observed how well additive models could do.

Logistic Regression + Additive Models

Several logistic regressions were developed using the Caret package. First, we used Recursive Feature Elimination (RFE) and 10-fold Cross-Validation to exhaustively test all combinations of parameters in a logistic regression. RFE is Caret's way of allowing for best subset selection, with a wider range of models than the linear regression subset functions. RFE produced a logistic model with the features shown below. The 1-SE rule was used to trade complexity for a small amount of standard deviation.

```
library(caret)
# Model creation using caret
train_control = trainControl(method="CV", number=10)

### LOGISTIC REGRESSION BEST SUBSET SELECTION USING CARET BACKWARD
### http://topepo.github.io/caret/recursive-feature-elimination.ht
x = twoyear.predictors %>% dplyr::select(-c("two_year_recid"))
#x <- x[train_ind, ]
y = as.factor(twoyear.predictors$two_year_recid)
#y <- y[train_ind]
#normalization <- preProcess(x)
#x <- predict(normalization, x)
x <- as.data.frame(x)

# Tell rfe what kind of model, how to perform cross-validation
ctrl <- rfeControl(method = "repeatedcv",
                   repeats = 10,
                   verbose = FALSE,
                   functions = lrFuncs)

# Cross-validate to find best model with each # of predictors incl
best_logistic_model = rfe(x[train_ind,], y[train_ind], sizes = sec
  metric = ifelse(is.factor(y[train_ind]), "Accuracy", "RMSE"),
  maximize = ifelse(is.factor(y[train_ind]), TRUE, FALSE),
  rfeControl = ctrl,
```

```
  method="glmnet")

plot(best_logistic_model, type = c("g", "o"))
```




```
best_logistic_model_1se = update(best_logistic_model,x[train_ind,
```

```
## Warning in update.rfe(best_logistic_model, x[train_ind, ], y[tr
## The saved resamples are no longer appropriate and were removed
```

```
# Outputs
best_logistic_pred = predict(best_logistic_model_1se,newdata = x[-
best_logistic_cm = caret::confusionMatrix(positive = "1", best_log
best_logistic_prob = predict(best_logistic_model_1se,newdata = x[-
logistic_roc = roc(y[-train_ind], best_logistic_prob)

best_logistic_model
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold, repeated 10
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.6316 0.2288    0.01750 0.03731
##          2   0.6755 0.3338    0.01794 0.03712
##          3   0.6793 0.3433    0.01793 0.03724
##          4   0.6806 0.3460    0.01802 0.03714        *
##          5   0.6784 0.3430    0.01922 0.03948
##          6   0.6768 0.3400    0.01895 0.03882
##          7   0.6762 0.3389    0.01847 0.03779
##          8   0.6760 0.3387    0.01800 0.03687
##          9   0.6754 0.3376    0.01832 0.03757
##         10   0.6754 0.3375    0.01835 0.03759
##
## The top 4 variables (out of 4):
##    priors_count, age, sex_Male, length_of_stay
```

Next, a logistic regression was developed using the lasso technique with 10-fold cross validation. In Caret, Lasso has internal feature selection, so the methods automatically provided an optimum alpha and lambda value of alpha = 1 and lambda = 0.002844731.

The figure above shows the Recursive Feature Elimination (RFE) plot for logistic regression as the number of variables vs. Accuracy. While peak Accuracy occurred at 6, this modeled settled on 2 due to the 1-SE rule.

```
### Logistic Regression using Lasso (has built-in feature selectic
lasso_logistic_model = caret::train(two_year_recid ~ ., trControl=
                            family="binomial", metric="Accuracy",

#plot(lasso_logistic_model, type = c("g", "o"))
# Should do 1-se but currently doesn't
```

```
# Outputs
lasso_logistic_pred = predict(lasso_logistic_model,newdata = x[-tr
lasso_logistic_cm = caret::confusionMatrix(positive = "1", lasso_l
lasso_logistic_prob = predict(lasso_logistic_model,newdata = x[-tr
lasso_roc = roc(y[-train_ind], lasso_logistic_prob)
```

Finally, an additive spline model logistic model was developed by fitting the parameters with a regular additive model, and graphically inspecting each parameter for higher-order relationships. Due to this inspection, 'age', 'priors_count', and 'length_of_stay' were chosen as 3rd and 4th degree spline terms for a logistic regression including all other parameters.

```
### Spline Fit Logistic Regression ###
library(gam)
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.16
```

```
spline.gam = gam(data=twoyear.predictors[train_ind, ],
                 formula = two_year_recid ~ s(age, 4) + s(priors_c

# Use to pick spline terms and order
# par(mfrow = c(10,2))
# plot(spline.gam, col="ForestGreen")

spline2.gam = gam(data=twoyear.predictors[train_ind, ],
                  formula = two_year_recid ~ . + s(age, 4) + s(pri

#Outputs
spline_pred = as.factor(as.integer(predict(spline2.gam,
                                           x[-train_ind, ], type =
```

```
spline_pred_cm = caret::confusionMatrix(positive = "1", spline_pre
spline_prob = predict(spline2.gam, x[-train_ind, ], type = "respor
spline_roc = roc(y[-train_ind], spline_prob)
```

KNN

KNN A K-Nearest Neighbors Model was developed using the Caret package, with 10-fold CV and all predictors included. Using Caret's auto-feature selection, the model was chosen with k = 43, with a yield of 0.67 Accuracy. While a KNN model was not ultimately used for our chosen, the 1-SE rule shows that we could have used k as low as 7 with similar results.

```
### K-NN ###
best_knn_model = caret::train(two_year_recid ~ ., data =twoyear.pr
                      method="knn", trControl=train_control, tune

# Outputs
knn_pred = predict(best_knn_model, newdata = x[-train_ind, ], type
knn_pred_cm = caret::confusionMatrix(positive = "1", knn_pred, y[-
knn_prob = predict(best_knn_model, newdata = x[-train_ind, ], type
knn_roc = roc(y[-train_ind], knn_prob)
```

For each of these logistic and Caret models, ROC curves show that the models perform very similarly, though no model is fully dominant- some have better Sensitivity at low cutoffs, while others trade for higher values at higher cutoffs.

Tree Methods

The first tree-based method that we decided to build is a simple pruned tree. Trees are arguably the most interpretable classification models and therefore could be very useful in the context of the justice system. Judges and defendants both would know exactly why someone is classified the way they are.

```
library(rpart)
library(partykit)
```

```
## Warning: package 'partykit' was built under R version 3.5.3
```

```
## Loading required package: grid
```

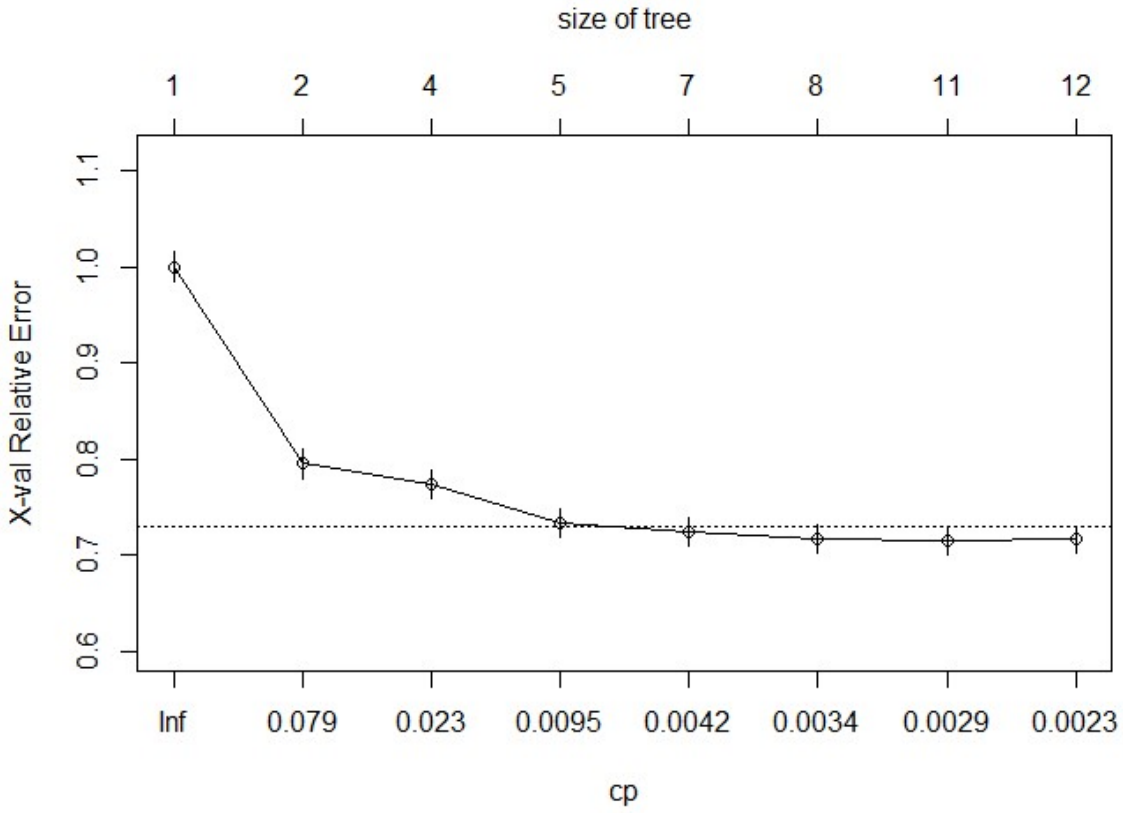
```
## Loading required package: libcoin
```

```
## Warning: package 'libcoin' was built under R version 3.5.3
```

```
## Loading required package: mvtnorm
```

```
## Warning: package 'mvtnorm' was built under R version 3.5.2
```

```
# From HW5

#tree.twoyear <- rpart(formula = two_year_recid ~ ., data = df.two
#party.twoyear <- as.party(tree.twoyear)

# Growing out large tree
twoyear.tree.full <- rpart(formula = two_year_recid ~ ., data = df
                           subset = train_ind,
                           control = rpart.control(minsplit=100, c
plotcp(twoyear.tree.full)
```




```
# Using one SE rule to determine what value of cp to use
One.SE <- which.min(twoyear.tree.full$cptable[, "xerror"])
One.SE.err <- twoyear.tree.full$cptable[One.SE, "xerror"] +
  twoyear.tree.full$cptable[One.SE, "xstd"]

(One.SE.index <- min(which(twoyear.tree.full$cptable[, "xerror"] <
```

```
## [1] 5
```

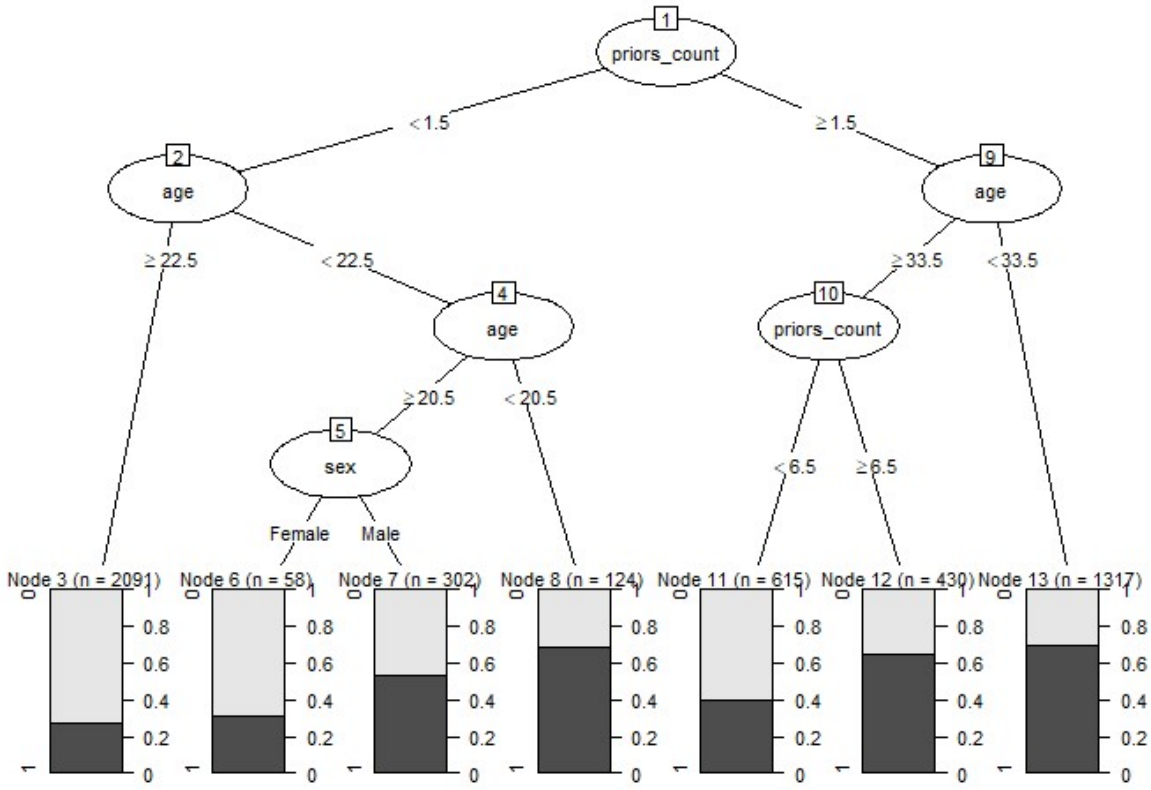
```
(One.SE.cp <- twoyear.tree.full$cptable[One.SE.index, "CP"])
```

```
## [1] 0.003544528
```

First, we use the rpart library to grow out a large tree. Next, we use the 1-se rule to determine the least complex model that we can realistically use. Above is a plot of error vs cp - the dashed line represents the 1-SE error.

We use our chosen value of cp to prune back our tree. Next, we used the partykit library to build a visualization of this pruned tree. We can see that the tree only uses priors and age to determine probabilities of recidivism.

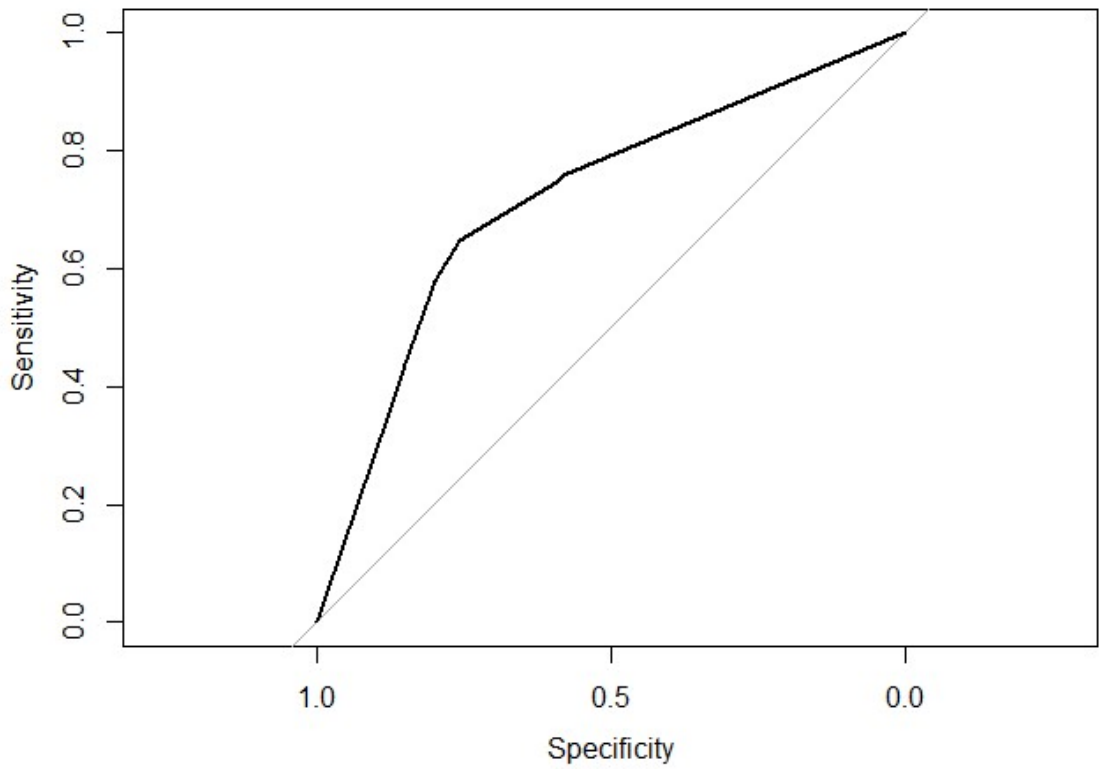
```
twoyear.pruned <- prune(tree = twoyear.tree.full, cp = One.SE.cp)
twoyear.pruned.party <- as.party(twoyear.pruned)

plot(twoyear.pruned.party, gp = gpar(fontsize = 8))
```




```
pruned.predictions <- predict(twoyear.pruned,
                              newdata = df.twoyear[-train_ind,], t
y <- df.twoyear[-train_ind, "two_year_recid"]
pruned_cm <- caret::confusionMatrix(positive = "1", pruned.predict
```

```
pruned.probabilities <- predict(twoyear.pruned,
                                newdata = df.twoyear[-train_ind,])
pruned.roc <- roc(response = y,
```

```
                predictor = pruned.probabilities)
plot(pruned.roc)
```



Random Forest

Next we build a random forest model. Random forests aren't easily interpretable, but for certain problems they can perform very well. We tried two models, varying the number of variables to split on between 2 and 3. We then plotted the Out of Bag Error vs Number of trees for both models. We observed that the model that allows 2 variables per split outperforms the one that allows 3 per split. We also observed that by around 300 trees, the error has largely leveled out; therefore, we choose the number of trees minimizing OOB Error that is under 300 trees.

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

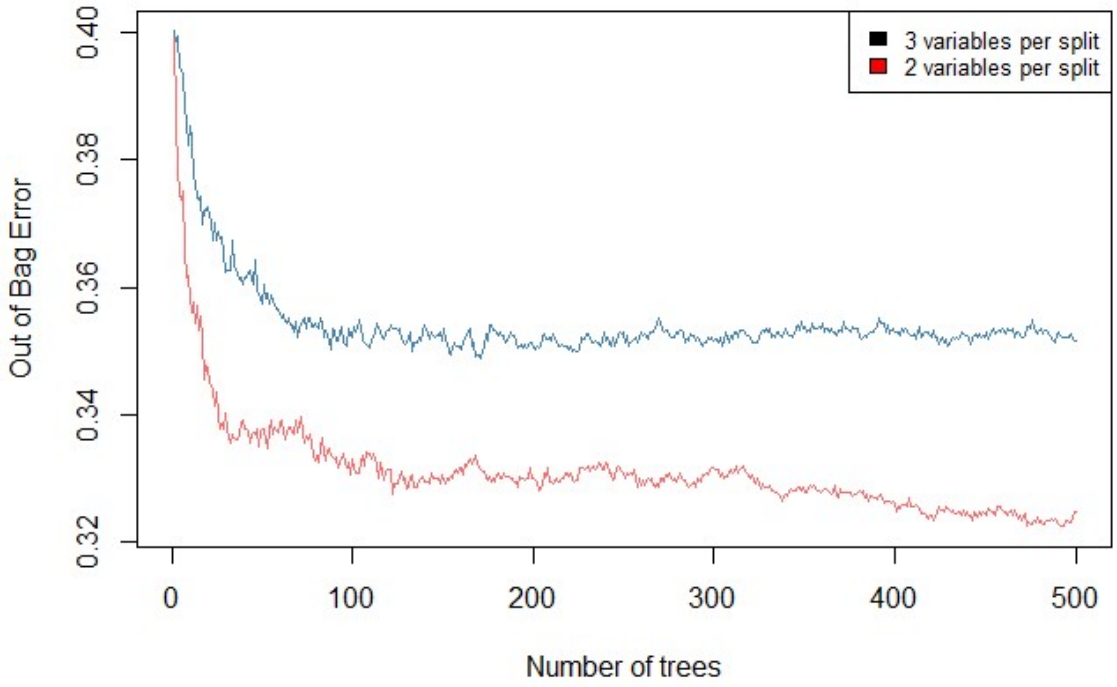
```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
rf.twoyear.2 <- randomForest(two_year_recid ~ .,
                             data = df.twoyear[train_ind,],
                             mtry = 2 ,importance = TRUE)
rf.twoyear.3 <- randomForest(two_year_recid ~ .,
                             data = df.twoyear[train_ind,],
                             mtry = 3,importance = TRUE)

plot(rf.twoyear.2$err.rate[,1], type = "line", col = "indianred1",
      xlab="Number of trees", ylab="Out of Bag Error")
lines(rf.twoyear.3$err.rate[,1], col = "steelblue")
legend("topright", c("3 variables per split", "2 variables per spl
       cex=0.8, fill=1:4)
```




```
# Choose 2 variables per split tree with 150 trees
rf.twoyear <- randomForest(two_year_recid ~ .,
                           data = df.twoyear[train_ind,],
                           mtry = 2,
                           ntree = which.min(rf.twoyear.2$err.rate
                           importance = TRUE)
```

We use importance plots to determine which predictors are contributing the most to outcomes. With mean decrease in accuracy, we see that age and priors_count are the most impactful. With mean decrease in Gini, we see that length_of_stay also has a large impact.

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.3
```

```
## randomForest 4.6-14
```

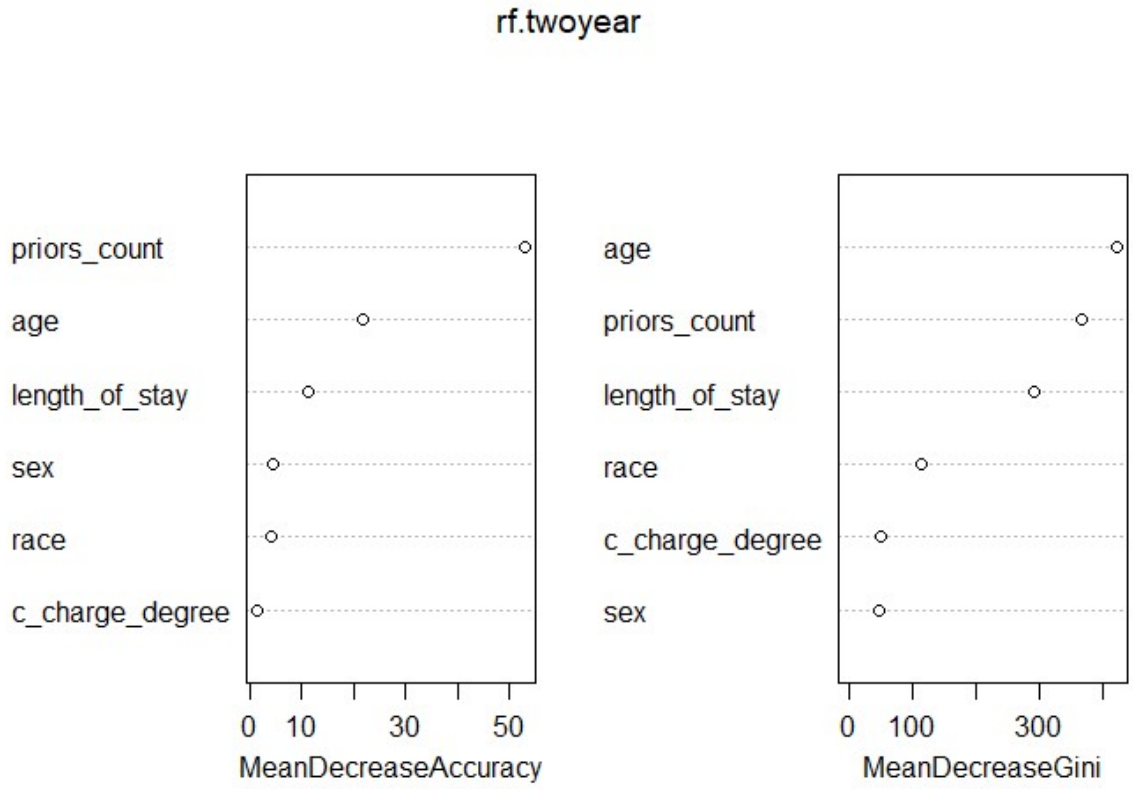
```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

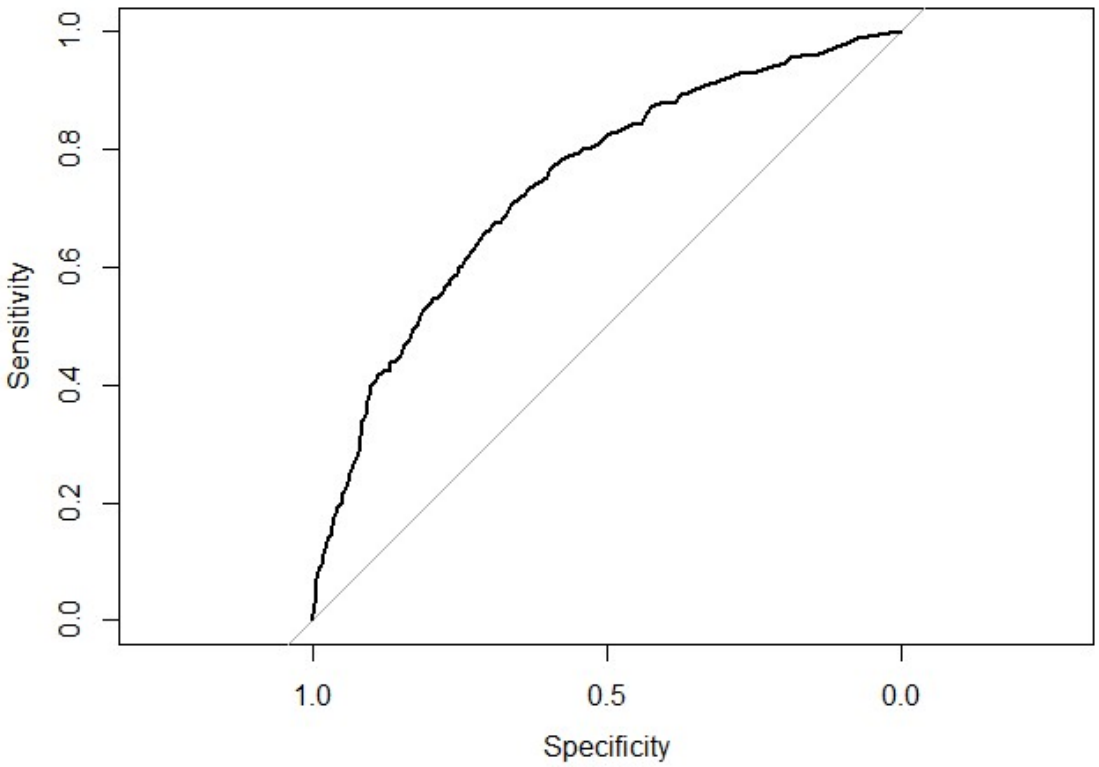
```
varImpPlot(rf.twoyear)
```



** Most impactful predictors seem to be age, priors, and length of stay **

```
rf.predictions <- predict(rf.twoyear,
                          newdata = df.twoyear[-train_ind,],
                          type = "class")
y <- df.twoyear[-train_ind, "two_year_recid"]
rf_cm <- caret::confusionMatrix(positive = "1", rf.predictions, y)
```

```
rf.probabilities <- predict(rf.twoyear, newdata = df.twoyear[-trai
                            type = "prob")[,2]
rf.roc <- roc(response = y,
              predictor = rf.probabilities)
plot(rf.roc)
```



Analyzing Our Models

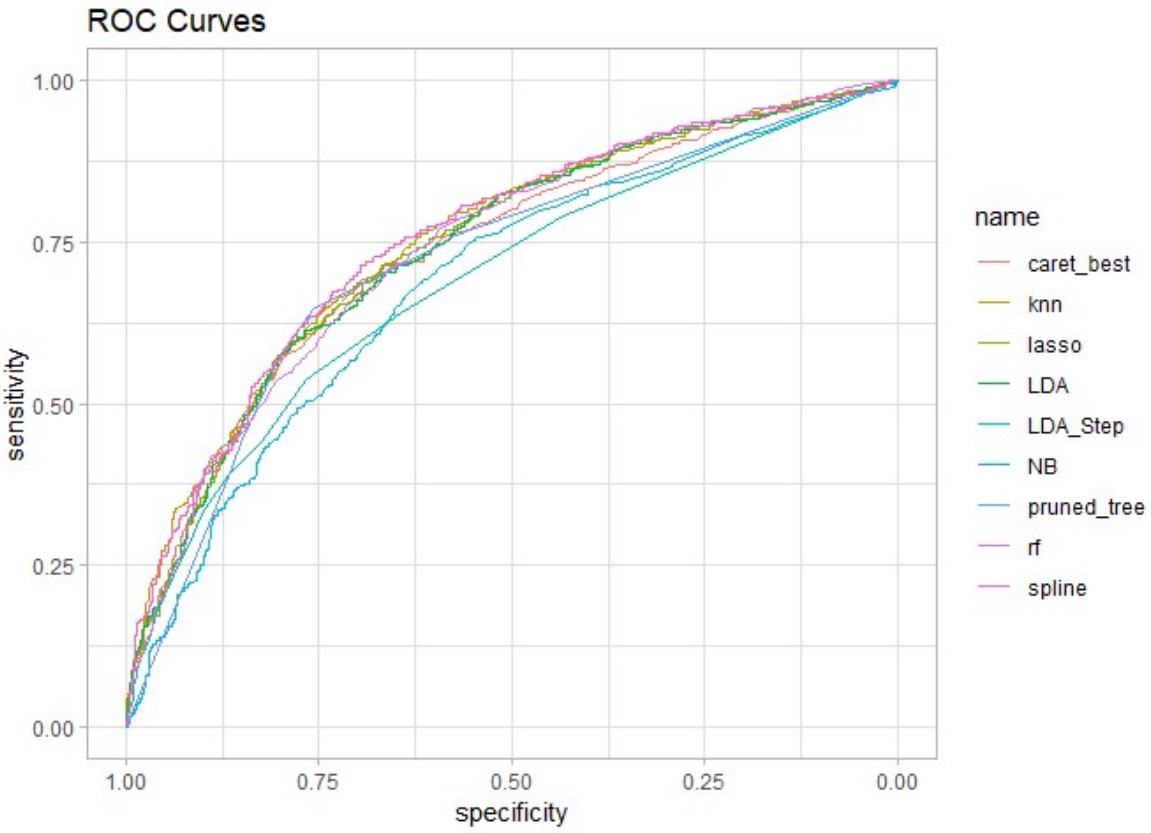
Underlying our positioning and framing of the Risk Assessment Instrument was a question of bias and tradeoffs. We built our assessment around the following performance metric equations:

- Sensitivity = TP/(TP + FN) Higher Sensitivity represents better predicting a higher # of those who recidivate.
- Specificity = TN/(TN + FP) Higher specificity represents better not predicting those who ultimately do not recidivate.
- Positive Predictive Value = TP/(TP + FP) Higher PPV represents an improved quality of prediction, with a higher % of our recidivation predictions being correct.

Philosophically, the trade-offs between these performance metrics may mean the difference between wrongly sentencing a person unlikely to recidivate, as well as lightly sentencing someone who may be very likely to recidivate. This is a historic debate had in law schools and societies.

Recognizing the prevalence of racial bias in tools, we aimed to maximize each Sensitivity, Specificity, and PPV, with priorities on Specificity and PPV. We plotted the ROC curves of our model to look for clear winners in term of AUC. Unfortunately, most of our models performed similarly well. While it was possible to weed out models like Naive Bayes and the LDA-step function, we observed no clear cut victors.

```
ggroc(list(NB = nb_roc, LDA = lda_roc, LDA_Step = lda_stepwise_roc
```



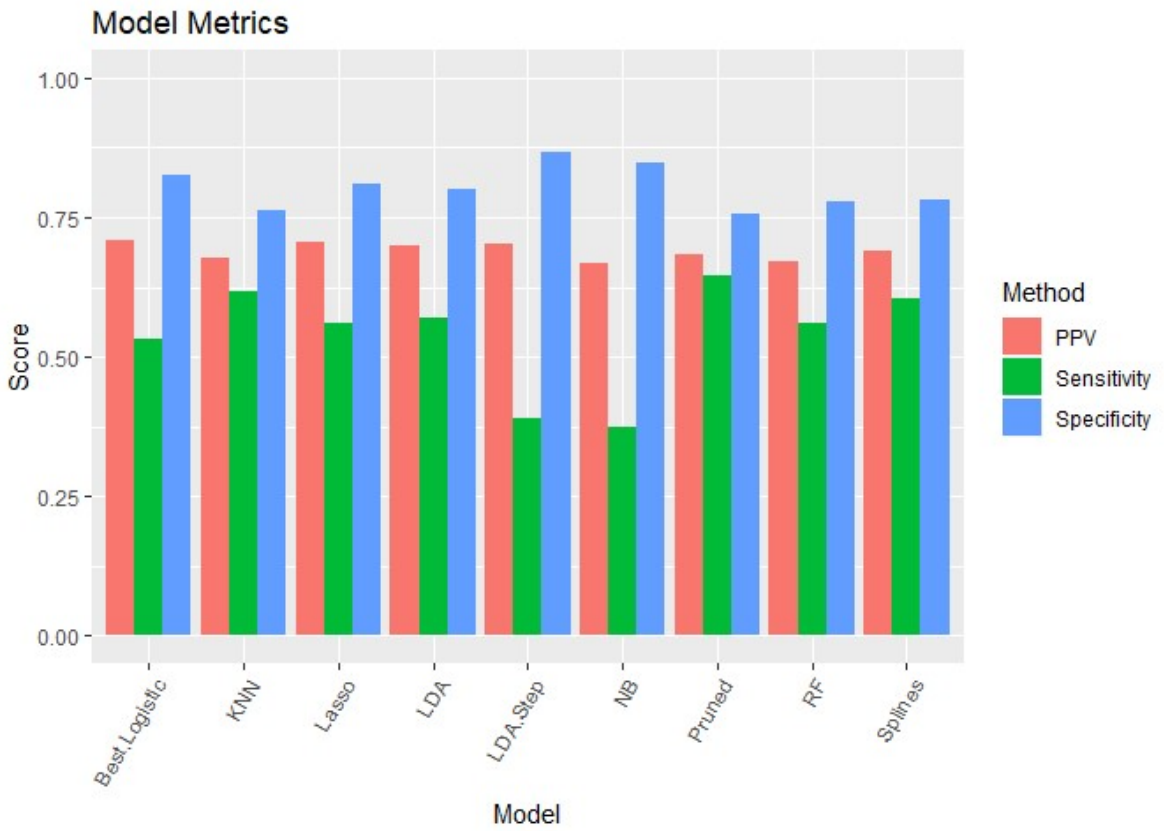
Next, to examine each performance metric, we generated a dataframe and bar graph to map out how each model performed.

Building a dataframe

```
performance.metrics <- data.frame("Best Logistic" = best_logistic_
                                  "Lasso" = lasso_logistic_cm$byCl
                                  "LDA Step" = lda_stepwise_cm$byC
                                  "RF" = rf_cm$byClass, "Splines"
performance.metrics <- rbind(performance.metrics, c(logistic_roc$a
                                                    lda_stepwise_r
                                                    rf.roc$auc, sp
performance.metrics <- t(performance.metrics)
colnames(performance.metrics)[12] <- "AUC"
performance.metrics <- as.data.frame(performance.metrics)
performance.metrics <- performance.metrics %>% dplyr::select("AUC"
performance.metrics <- cbind(performance.metrics, rownames(perform
colnames(performance.metrics)[13] <- "Model"
```

Visualizing Results

```
perf.sens <- performance.metrics %>% dplyr::select("Sensitivity",
perf.spec <- performance.metrics %>% dplyr::select("Specificity",
perf.PPV <- performance.metrics %>% dplyr::select("Pos Pred Value"
colnames(perf.sens) <- c("Score", "Model", "Method")
colnames(perf.spec) <- c("Score", "Model", "Method")
colnames(perf.PPV) <- c("Score", "Model", "Method")

tradeoff.data <- data.frame(rbind(perf.sens, perf.spec, perf.PPV))

# Run for violent and save plot
ggplot(data = tradeoff.data) +
  geom_bar(mapping = aes(x = Model, y = Score, fill = Method),
           position = "dodge", stat = "identity") +
  ylim(c(0,1))+
  theme(axis.text.x = element_text(angle = 60, hjust = 1))+
  ggtitle("Model Metrics")
```

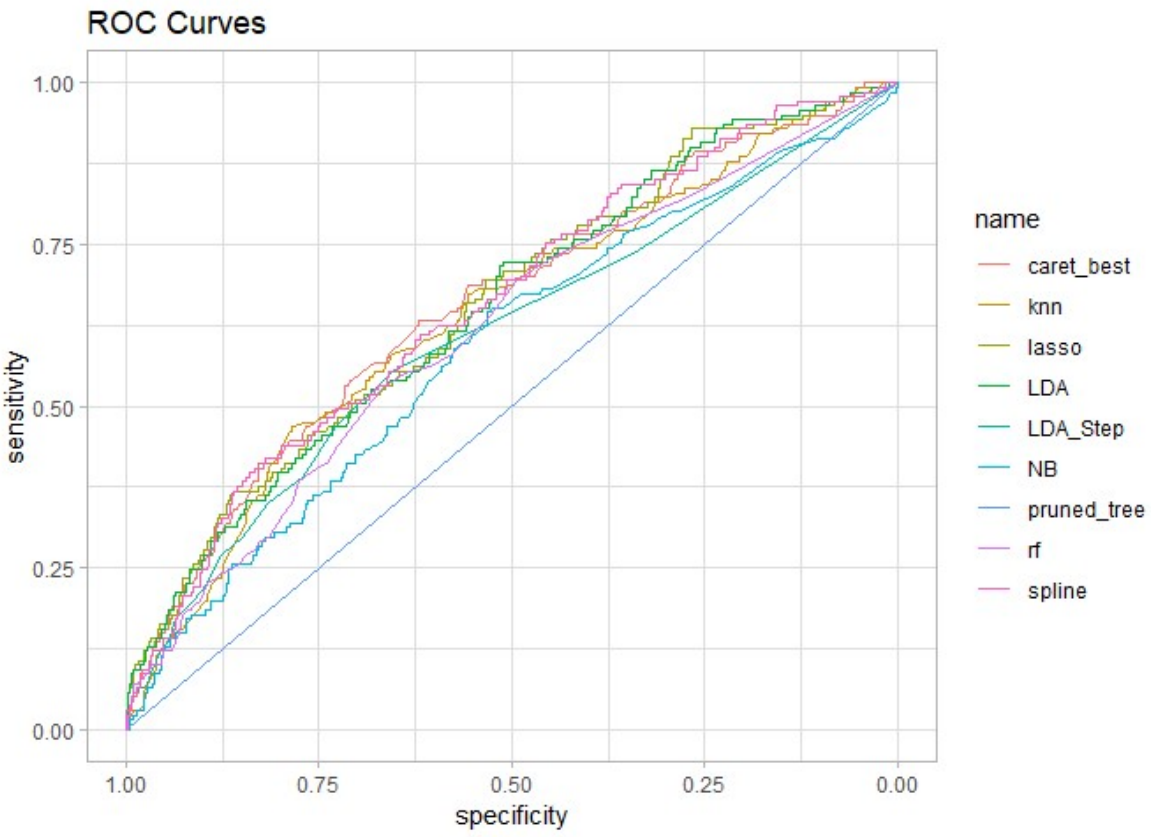


Best Model: General Recidivism

Upon witnessing these results, we observed that a pruned forest model performed admirably on many key metrics. It scored highest on metrics of sensitivity (0.658), while maintaining a competitive PPV (0.68). Thus, it became our strongest model for two-year recidivism. We also chose the tree for its interpretability, which inherently makes risk assessment seem more fair.

We ran the same plots for violent recidivism:

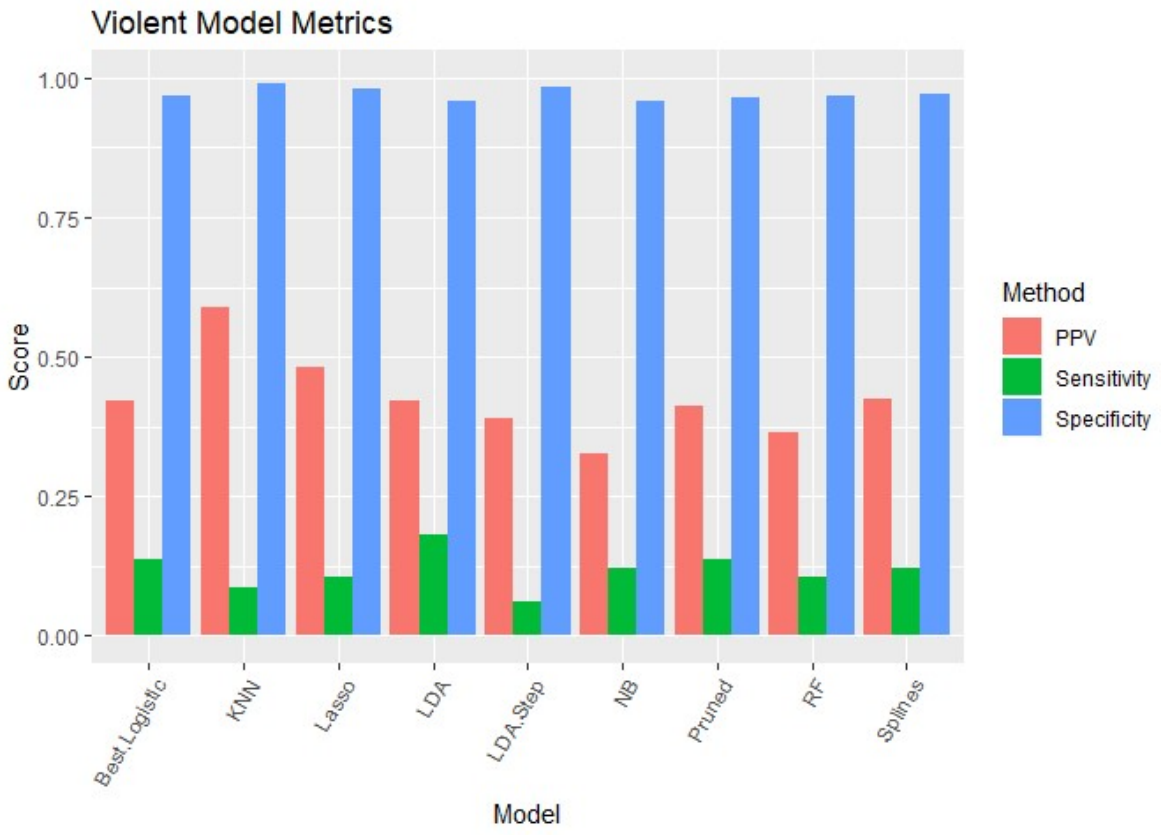
```
load("./violent_ROC_plot.RData")
violent_ROC_plot
```



And produced similar visualizations:

```
#current = getwd()
library(caret)
library(dplyr)
```

```
load(file = "C:/Users/bjrhi/OneDrive/Documents/R/DataMining/final_

performance.metrics.violent <- cbind(performance.metrics.violent,
performance.metrics.violent <- as.data.frame(performance.metrics.v
colnames(performance.metrics.violent)[12] <- "Model"
perf.sens.v <- performance.metrics.violent %>% dplyr::select("Sens
perf.spec.v <- performance.metrics.violent %>% dplyr::select("Spec
perf.PPV.v <- performance.metrics.violent %>% dplyr::select("Pos P
colnames(perf.sens.v) <- c("Score", "Model", "Method")
colnames(perf.spec.v) <- c("Score", "Model", "Method")
colnames(perf.PPV.v) <- c("Score", "Model", "Method")

tradeoff.data.v <- data.frame(rbind(perf.sens.v, perf.spec.v, perf
tradeoff.data.v$Score <- as.numeric(levels(tradeoff.data.v$Score))
tradeoff.data.v$Model <- as.factor(tradeoff.data.v$Model)
tradeoff.data.v$Method <- as.factor(tradeoff.data.v$Method)

ggplot(data = tradeoff.data.v) +
  geom_bar(mapping = aes(x = Model, y = Score, fill = Method),
           position = "dodge", stat = "identity") +
  ylim(c(0,1))+
  theme(axis.text.x = element_text(angle = 60, hjust = 1))+
  ggtitle("Violent Model Metrics")
```



Best Model: Violent General Recidivism

Out of 4015 separate observations in the violent recidivism dataset, there is only a prevalence of 4.4%. Thus, we are working with a data set with mostly non-recidivation. For this reason, at a 0.5 cutoff, our default for model comparison, most models have a very high Specificity with low Sensitivity. This is a fair bet, but means we are likely to under-classify those who are likely to violently recidivate. (Note: while Pro-Publica used a variation on the full two-year data-set, in the cleaning process we had data-integrity issues and decided to use the two-year-compass-violent.csv data set, which has fewer points).

Despite this limitation, we found that the LDA model best compromised between high quality results with its 0.42 PPV and pack-leading 0.18 Sensitivity.

Between Model Comparisons

Assessing Final Models: General Recidivism

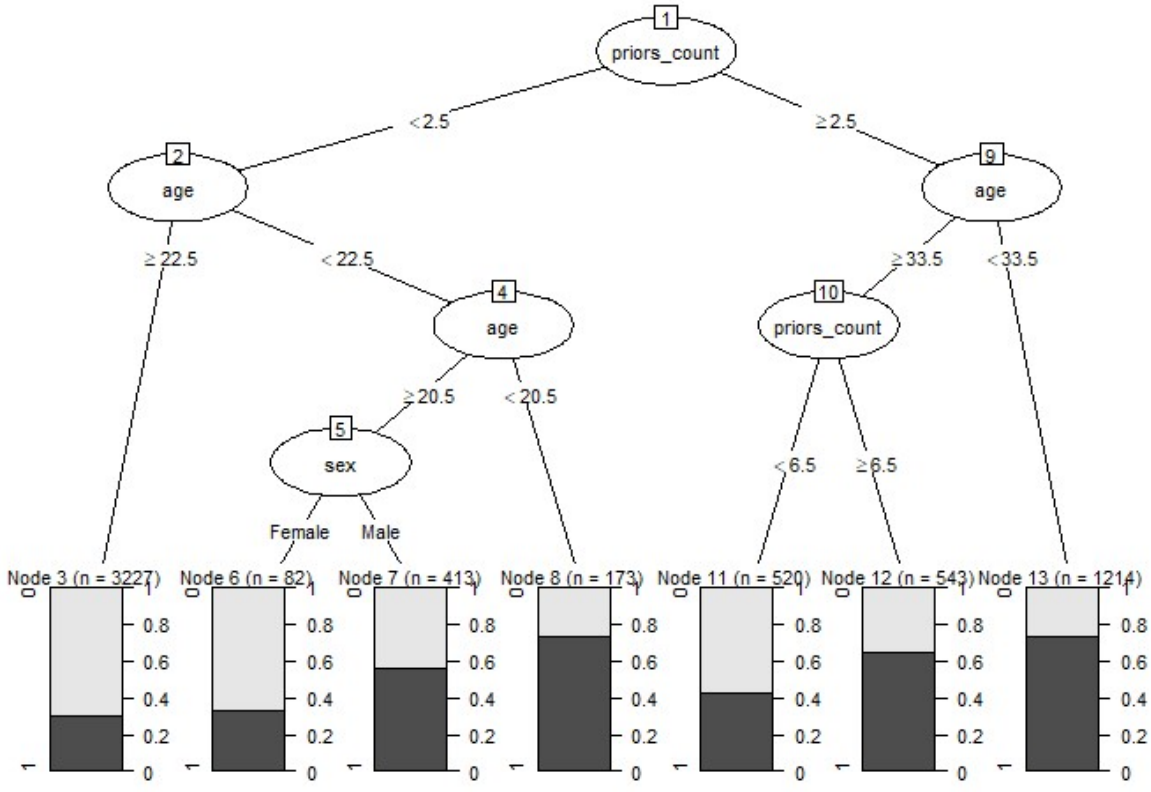
Once we selected the Pruned Tree model for general recidivism and LDA for violent recidivism, we developed a Cross-Validation method (10-Fold) to test each data point against both models. Data was randomly divided into 10 folds, and 10-models (90% train, 10% hold out) were constructed and tested against the hold out sets. This method allowed us to assess accuracy without over-predicting by training and testing on the full data set.

```
## Warning: package 'partykit' was built under R version 3.5.3
```

```
## Warning: package 'libcoin' was built under R version 3.5.3
```

```
## Warning: package 'mvtnorm' was built under R version 3.5.2
```

```
big.tree <- rpart(formula = two_year_recid ~ ., data = df.twoyear,
                        control = rpart.control(minsplit = 100, cp

best.pruned.tree <- prune(tree = big.tree, cp = chosen.cp)

best.tree.party <- as.party(best.pruned.tree)
plot(best.tree.party, gp = gpar(fontsize = 8))
```



Our final pruned tree uses three variables to predict two year recidivism: priors_count, age, and sex. In general, the more priors you have, the younger your age, and being make all make you more likely to recidivate. If someone has more than 2.5 priors and is under 33.5 years old, they have about a 70% chance of recidivating within two years.

Assessing Final Models: Violent Recidivism

The most important positive predictors for LDA were: number of priors, race, age, and sex. The below model compares each race to a baseline of being black.

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(klaR)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
best.model.v <- lda_fit

classificationCrossValidation <- function(df.twoyear.violent) {

  # Standardization
  standardized_params = preProcess(df.twoyear.violent, method=c("c
  standardized_data = predict(standardized_params,df.twoyear.viole

  # 10-fold Cross-Validation to test for Classification Error
  random_indecies = sample(1:(nrow(df.twoyear.violent)))
  start = 1
  fold_size = floor(nrow(df.twoyear.violent)/10)
  final_2y_scores = data.frame()
  final_2y_predics = c()
  y = c()
  indics = data.frame()

  for(i in seq(1,10)){

    if(i==10){
      range = seq(start + (i-1)*fold_size,nrow(df.twoyear.violent)
    } else{
      range = seq(start + (i-1)*fold_size,i*fold_size)
    }

    test_vector = random_indecies[range]

    fold_train = standardized_data[-test_vector,]
```

```
    fold_test = standardized_data[test_vector,]

    lda_fit <- lda(two_year_recid ~., data=fold_train)

    x <- as.data.frame(fold_test %>% dplyr::select(-c("two_year_re
    y <- c(y, fold_test$two_year_recid)
    indics <- rbind(indics,df.twoyear.violent[test_vector,])

    # put final_2y_predics, y, indics into a daa frame together

    final_2y_scores <- rbind(final_2y_scores,data.frame(predict(lc
    final_2y_predics <- as.factor(as.integer(final_2y_scores$poste
  }

  y = as.factor(y-1)

  return(data.frame(predics = final_2y_predics, y = y, indics))

}

load(file="./df.twoyear.violent.RData")

# use on violent data set
crossVals = classificationCrossValidation(df.twoyear.violent)
# Confusion Matrix for Cross-Validated Model
best.cm.violent <- caret::confusionMatrix(positive = "1", crossVal
```

```
##                            LD1
## priors_count          0.81362405
## raceAsian            -0.75541263
## age                  -0.61095683
## raceNative American -0.46139721
## sexMale               0.43324632
## raceOther            -0.32987988
## raceHispanic         -0.32162898
## c_charge_degreeM     -0.26612389
## length_of_stay        0.13496816
## raceCaucasian        -0.08863264
```

Comparing Results to COMPAS

Our best models selected, we began to compare our results to COMPAS. Here is how COMPAS performed, according to Propublica's research. For all defendants, this is what they observed:

Total: 7214.00 False positive rate: 32.35 *False negative rate: 37.40* Specificity: 0.68 *Sensitivity: 0.63* Prevalence: 0.45 *PPV: 0.61* NPV: 0.69

Our General Recidivism model performed overall with a 68.4% Accuracy. Compared to COMPAS it was 10-points better (lower) on overall False Positive Rates and 5.5-points worse (higher) on False Negatives. From our perspective of reducing bias, so far, so good, but at the price of not classifying more recidivators.

```
## [1] "COMPAS False Positive Rate: 32.35 | Our models False Posit
```

```
## [1] "COMPAS False Positive Rate: 37.40 | Our models False Negat
```

```
## [1] "-------------------------- "
```

```
## [1] "Violent False Positive Rate: 0.22 | Our models False Negat
```

Our Violent Recidivism model on the other hand was actually more accurate- 83.9%. Why? Because with a smaller dataset, with a much smaller prevalence, the LDA was more likely to generate a negative assignment. Thus, resulting in an 86.04% False Negative Rate, and a just a 2.5% False Positive Rate. Compare this to COMPAS' 27.93% False Positive Rate and a 47.15% False Negative Rate.

```
## [1] "White Confusion Matrix"
```

```
## Confusion Matrix and Statistics
##
```

```
##          Reference
## Prediction    0    1
##          0 1065  468
##          1  216  354
##
##               Accuracy : 0.6748
##                 95% CI : (0.6543, 0.6948)
##    No Information Rate : 0.6091
##    P-Value [Acc > NIR] : 2.553e-10
##
##                  Kappa : 0.2773
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4307
##             Specificity : 0.8314
##          Pos Pred Value : 0.6211
##          Neg Pred Value : 0.6947
##              Prevalence : 0.3909
##          Detection Rate : 0.1683
##    Detection Prevalence : 0.2710
##       Balanced Accuracy : 0.6310
##
##        'Positive' Class : 1
##
```

```
## [1] "White Confusion-Violence Matrix"
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction    0    1
##          0 1916  186
##          1    1    0
##
##               Accuracy : 0.9111
##                 95% CI : (0.8981, 0.9229)
##    No Information Rate : 0.9116
##    P-Value [Acc > NIR] : 0.55
```

```
##
##                  Kappa : -0.0009
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000000
##             Specificity : 0.9994784
##          Pos Pred Value : 0.0000000
##          Neg Pred Value : 0.9115128
##              Prevalence : 0.0884451
##          Detection Rate : 0.0000000
##    Detection Prevalence : 0.0004755
##       Balanced Accuracy : 0.4997392
##
##        'Positive' Class : 1
##
```

```
## [1] "African American Confusion Matrix"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1014  565
##          1  500 1096
##
##                Accuracy : 0.6646
##                  95% CI : (0.6478, 0.681)
##     No Information Rate : 0.5231
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.329
##
##  Mcnemar's Test P-Value : 0.04986
##
##             Sensitivity : 0.6598
##             Specificity : 0.6697
##          Pos Pred Value : 0.6867
##          Neg Pred Value : 0.6422
```

```
##              Prevalence : 0.5231
##          Detection Rate : 0.3452
##    Detection Prevalence : 0.5027
##       Balanced Accuracy : 0.6648
##
##        'Positive' Class : 1
##
```

```
## [1] "African American-Violent Confusion Matrix"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2740  421
##          1    9    5
##
##                Accuracy : 0.8646
##                  95% CI : (0.8522, 0.8763)
##     No Information Rate : 0.8658
##     P-Value [Acc > NIR] : 0.595
##
##                   Kappa : 0.0143
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.011737
##             Specificity : 0.996726
##          Pos Pred Value : 0.357143
##          Neg Pred Value : 0.866814
##              Prevalence : 0.134173
##          Detection Rate : 0.001575
##    Detection Prevalence : 0.004409
##       Balanced Accuracy : 0.504232
##
##        'Positive' Class : 1
##
```

```
## [1] "Caucasian False Positive Rate: 16.86"
```

```
## [1] "African American False Positive Rate: 33.03"
```

```
## [1] "Caucasian False Negative Rate: 56.93"
```

```
## [1] "African American False Negative Rate: 34.02"
```

Our model has higher False Positive Rates for white than black individuals. It has higher False Negative Rates for white individuals too. Our model seems to not apply to races equally.

```
## [1] "Violent Caucasian False Positive Rate: 0.05"
```

```
## [1] "Violent African American False Positive Rate: 0.33"
```

```
## [1] "Violent Caucasian False Negative Rate: 100"
```

```
## [1] "Violent African American False Negative Rate: 98.83"
```

Our violent recidivism model has higher FPR for African Americans and lower FNR. This is considered racially biased as well.

```
## [1] "COMPAS Black False Positive Rate: 44.85 | Our model's Blac
```

```
## [1] "COMPAS Black False Negative Rate: 27.99 | Our model's Blac
```

```
## [1] "COMPAS Black False Sensitivity: 0.72 | Our model's Black S
```

```
## [1] "COMPAS Black False Specificity: 0.55 | Our model's Black S
```

```
## [1] "----- RESULTS FOR VIOLENT RECIDIVISM -----"
```

```
## [1] "COMPAS Black False Positive Rate: 44.85 | Our model's Blac
```

```
## [1] "COMPAS Black False Negative Rate: 27.99 | Our model's Blac
```

```
## [1] "COMPAS Black False Sensitivity: 0.72 | Our model's Black S
```

```
## [1] "COMPAS Black False Specificity: 0.55 | Our model's Black S
```

Model performance for Sex and Age:

As shown by these models, under our tool, Males have a higher sensitivity than Females but a lower Specificity. As for age, Sensitivity decreases from Young - Old, possibly due to the lack of true positive points for older people. Meanwhile, Specificity increases, as it is more likely to correctly categorize True (and False) Negatives.

As shown before, our violent tool is systemically low in Sensitivity and high in Specificity. Thus, Age and Sex results take these conclusions to the extreme, with only adults and the elderly having lower than 99% False Negative Rates.

```
## [1] "----- RESULTS FOR  SEX AND AGE -----"
```

```
## [1] "----- MALE  -----"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1891  949
```

```
##          1  710 1447
##
##                Accuracy : 0.668
##                  95% CI : (0.6547, 0.6811)
##     No Information Rate : 0.5205
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3323
##
##  Mcnemar's Test P-Value : 5.12e-09
##
##             Sensitivity : 0.6039
##             Specificity : 0.7270
##          Pos Pred Value : 0.6708
##          Neg Pred Value : 0.6658
##              Prevalence : 0.4795
##          Detection Rate : 0.2896
##    Detection Prevalence : 0.4317
##       Balanced Accuracy : 0.6655
##
##        'Positive' Class : 1
##
```

```
## [1] "----- FEMALE  -----"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 664 251
##          1  98 162
##
##                Accuracy : 0.703
##                  95% CI : (0.6759, 0.729)
##     No Information Rate : 0.6485
##     P-Value [Acc > NIR] : 4.280e-05
##
##                   Kappa : 0.2881
##
```

```
##  Mcnemar's Test P-Value : 4.073e-16
##
##             Sensitivity : 0.3923
##             Specificity : 0.8714
##          Pos Pred Value : 0.6231
##          Neg Pred Value : 0.7257
##              Prevalence : 0.3515
##          Detection Rate : 0.1379
##    Detection Prevalence : 0.2213
##       Balanced Accuracy : 0.6318
##
##        'Positive' Class : 1
##
```

```
## [1] "----- YOUNG  -----"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 305 207
##          1 288 547
##
##                Accuracy : 0.6325
##                  95% CI : (0.6061, 0.6583)
##     No Information Rate : 0.5598
##     P-Value [Acc > NIR] : 0.00000003489
##
##                   Kappa : 0.2433
##
##  Mcnemar's Test P-Value : 0.0003235
##
##             Sensitivity : 0.7255
##             Specificity : 0.5143
##          Pos Pred Value : 0.6551
##          Neg Pred Value : 0.5957
##              Prevalence : 0.5598
##          Detection Rate : 0.4061
##    Detection Prevalence : 0.6199
```

```
##       Balanced Accuracy : 0.6199
##
##        'Positive' Class : 1
##
```

```
## [1] "----- ADULT  -----"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1479  747
##          1  412  894
##
##                Accuracy : 0.6719
##                  95% CI : (0.6561, 0.6873)
##     No Information Rate : 0.5354
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3314
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5448
##             Specificity : 0.7821
##          Pos Pred Value : 0.6845
##          Neg Pred Value : 0.6644
##              Prevalence : 0.4646
##          Detection Rate : 0.2531
##    Detection Prevalence : 0.3698
##       Balanced Accuracy : 0.6635
##
##        'Positive' Class : 1
##
```

```
## [1] "----- OLD  -----"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 771 246
##          1 108 168
##
##                Accuracy : 0.7262
##                  95% CI : (0.701, 0.7504)
##     No Information Rate : 0.6798
##     P-Value [Acc > NIR] : 0.0001638
##
##                   Kappa : 0.3103
##
##  Mcnemar's Test P-Value : 3.302e-13
##
##             Sensitivity : 0.4058
##             Specificity : 0.8771
##          Pos Pred Value : 0.6087
##          Neg Pred Value : 0.7581
##              Prevalence : 0.3202
##          Detection Rate : 0.1299
##    Detection Prevalence : 0.2135
##       Balanced Accuracy : 0.6415
##
##        'Positive' Class : 1
##
```

```
## [1] "----- RESULTS FOR VIOLENT SEX AND AGE -----"
```

```
## [1] "----- MALE VIOLENT -----"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4375  605
```

```
##          1   12    5
##
##                Accuracy : 0.8765
##                  95% CI : (0.8671, 0.8855)
##     No Information Rate : 0.8779
##     P-Value [Acc > NIR] : 0.6289
##
##                   Kappa : 0.0094
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.008197
##             Specificity : 0.997265
##          Pos Pred Value : 0.294118
##          Neg Pred Value : 0.878514
##              Prevalence : 0.122073
##          Detection Rate : 0.001001
##    Detection Prevalence : 0.003402
##       Balanced Accuracy : 0.502731
##
##        'Positive' Class : 1
##
```

```
## [1] "----- FEMALE VIOLENT -----"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1093   82
##          1    0    0
##
##                Accuracy : 0.9302
##                  95% CI : (0.9141, 0.9441)
##     No Information Rate : 0.9302
##     P-Value [Acc > NIR] : 0.5293
##
##                   Kappa : 0
##
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.93021
##              Prevalence : 0.06979
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 1
##
```

```
## [1] "----- YOUNG VIOLENT -----"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1161  185
##          1    1    0
##
##                Accuracy : 0.8619
##                  95% CI : (0.8423, 0.8799)
##     No Information Rate : 0.8627
##     P-Value [Acc > NIR] : 0.551
##
##                   Kappa : -0.0015
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000000
##             Specificity : 0.9991394
##          Pos Pred Value : 0.0000000
##          Neg Pred Value : 0.8625557
##              Prevalence : 0.1373422
##          Detection Rate : 0.0000000
##    Detection Prevalence : 0.0007424
```

```
##       Balanced Accuracy : 0.4995697
##
##        'Positive' Class : 1
##
```

```
## [1] "----- ADULT VIOLENT -----"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3099  421
##          1    8    4
##
##                Accuracy : 0.8785
##                  95% CI : (0.8673, 0.8891)
##     No Information Rate : 0.8797
##     P-Value [Acc > NIR] : 0.5944
##
##                   Kappa : 0.0118
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.009412
##             Specificity : 0.997425
##          Pos Pred Value : 0.333333
##          Neg Pred Value : 0.880398
##              Prevalence : 0.120328
##          Detection Rate : 0.001133
##    Detection Prevalence : 0.003398
##       Balanced Accuracy : 0.503418
##
##        'Positive' Class : 1
##
```

```
## [1] "----- OLD VIOLENT -----"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1208   81
##          1    3    1
##
##                Accuracy : 0.935
##                  95% CI : (0.9202, 0.9479)
##     No Information Rate : 0.9366
##     P-Value [Acc > NIR] : 0.6181
##
##                   Kappa : 0.0175
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0121951
##             Specificity : 0.9975227
##          Pos Pred Value : 0.2500000
##          Neg Pred Value : 0.9371606
##              Prevalence : 0.0634184
##          Detection Rate : 0.0007734
##    Detection Prevalence : 0.0030936
##       Balanced Accuracy : 0.5048589
##
##        'Positive' Class : 1
##
```

Closing Words

Ultimately, our findings are slightly better than the COMPAS tool with regards to True Positives and generally have fewer False Positives. In addition, our models are consistent with the wealth of criminal justice research. Ages, sex, and number of priors are consistently associated with likelihood to recidivate. (Richard Berk, Criminal justice forecasts of risk, (New York, Springer, 2002),30.)

Our analysis reveals that these tools can sometimes adequately predict

who may be likely to recidivate, but these decisions must be accompanied by a level of intentional fairness to ensure vulnerable populations are not adequately targeted.