DS 303 Final Project Recidivism Classifier

```
df = read.csv("data_cleaned.csv")
```

Logistic Regression Model

One question that the topic of the data is begging to be asked is if we can predict whether someone released will return to prison within the three year time frame of the data. First, let's see what proportion returned:

```
sum(df$Return.to.Prison) / length(df$Return.to.Prison)
```

```
## [1] 0.333628
```

So around a third of prisoners returned within three years.

Let's consider what values are useful for predicting Return.to.Prison. There are some variables we will need to consider altering, combining, or perhaps dropping, like Main.Supervising.District (MSD), which appears to be correlated with Release.Type (i.e., there is often no MSD when a prisoner is put on parole). Similar correlation is present in Target.Population. We will save these for later. Fiscal.Year.Released and Recidivism.Reporting.Year are 2010 and 2013 respectively for all inmates, since they are only labels of the dataset itself, so we will not use them for the model. Additionally, we are predicting whether recidivism will occur, so we will not include variables that confirm it, meaning any new-offense-related variables. These include: Days.to.Return, Recidivism.Type, New.Offense.Classification, New.Offense.Type, and New.Offense.Sub.Type. A non-null value for any of these would, of course, confirm recidivism. We will hold off on excluding or combining the potentially correlated variables, Main.Supervising.District, Release.Type, and Target.Population, for now. We can later perform Chi-squared tests of independence on them. We will tentatively include them in the model. Below we also drop the single row with the value Interstate Compact Parole in Release.Type, as it is the only row with that value.

Before we fit the model, we will remove some rows with very rare values. While it is possible that these values may be good predictors, they are very rare (1-2 instances), so will not be especially useful in predicting the vast majority of cases. They also tend to cause issues when splitting training and testing sets.

```
# drop rows with very rare values (<3)
length(df$Release.Type)
```

```
## [1] 26020
```

```
indices_to_remove = vector()
indices_to_remove = append(indices_to_remove, which(df$Release.Type=="Interstate Compact Parole"))
indices_to_remove = append(indices_to_remove, which(df$Age.At.Release==""))
indices_to_remove = append(indices_to_remove, which(df$Race...Ethnicity=="Black -"))
indices_to_remove = append(indices_to_remove, which(df$Race...Ethnicity=="N/A -"))
indices_to_remove = append(indices_to_remove, which(df$Offense.Classification=="Other Misdemeanor"))
indices_to_remove = append(indices_to_remove, which(df$Offense.Classification=="Sexual Predator Communi
indices_to_remove = append(indices_to_remove, which(df$Offense.Classification=="Other Felony (Old Code)
df = df[-c(indices_to_remove),]
length(df$Release.Type)
```

```
## [1] 26007
```

Now, let's look at a full model using all of our potentially useful variables:

```
# first, train test split using ratio of 6/4
set.seed(20)
train = sample(1:nrow(df),nrow(df)*0.6, replace=FALSE)
test = (-train)

# fit the model
model.fit = glm(Return.to.Prison~Main.Supervising.District+Release.Type+Race...Ethnicity+Age.At.Release
#summary(model.fit)
```

Uncomment the summary line if necessary. Here it is commented out because the output is very long. Note that there are many many coefficients. This is because every variable is either boolean or categorical, and most of the categorical variables have several categories, resulting in a huge number of coefficients. Now, let's look at our actual predictions.

```
model.prob = predict(model.fit, df[test,],type='response')
head(model.prob)
```

```
##         1         3         5         9        11        13
## 0.2289770 0.4477712 0.2119533 0.4242753 0.4493011 0.3924763
```

```
model.pred = rep(FALSE, nrow(df)-length(test))
model.pred[model.prob >0.5] = TRUE
# confusion matrix
tbl.log = table(model.pred, df[test,]$Return.to.Prison)
tbl.log
```

```
##
## model.pred FALSE TRUE
##      FALSE  6431 2866
##      TRUE    501  605
```

```
# error rate
1-mean(model.pred == df[test,]$Return.to.Prison)
```

```
## [1] 0.3236566
```

```
# false positives
tbl.log[1, 2] / (tbl.log[1, 2] + tbl.log[2, 2])
```

```
## [1] 0.8256986
```

```
# false negatives
tbl.log[2, 1] / (tbl.log[2, 1] + tbl.log[1, 1])
```

```
## [1] 0.07227351
```

So, we have an error rate of around 32%, not terrible. What is terrible is our false positive rate, whereas we have a very low false negative rate. In this case, a false positive means incorrectly classifying a former inmate as one who will be rearrested. This may be seen as preferable to a false negative by, say, law enforcement who would rather be safe than sorry. A former inmate may prefer a false negative if a false positive means a stricter parole, more job restrictions, higher restitution, etc. It is not clear cut here what we should aim for, or what the consequences are for either type of error in practice. Next, we will see if we can improve the model using stepwise selection.

Let us the stepAIC from the MASS package to perform both forwards and backwards selection on our full model in an attempt to find a better one. Note that stepAIC uses AIC to make its decision.

```
library(MASS)

model.step = stepAIC(model.fit, trace=FALSE)
coef(model.step)
```

```
##                                                 (Intercept)
##                                                -25.12027761
##                                  Main.Supervising.District1JD
##                                                  1.47566062
##                                  Main.Supervising.District2JD
##                                                  1.51770049
##                                  Main.Supervising.District3JD
##                                                  1.50521980
##                                  Main.Supervising.District4JD
##                                                  1.57623146
##                                  Main.Supervising.District5JD
##                                                  1.71452671
##                                  Main.Supervising.District6JD
##                                                  1.52405104
##                                  Main.Supervising.District7JD
##                                                  1.24031670
##                                  Main.Supervising.District8JD
##                                                  1.38646690
##                      Main.Supervising.DistrictInterstate Compact
##                                                 -0.19681586
##                                  Main.Supervising.DistrictISC
##                                                 -0.13293299
##                 Release.TypeDischarged - Expiration of Sentence
##                                                 -0.13352800
##                      Release.TypeDischarged â\200" End of Sentence
##                                                  0.28920341
##                                       Release.TypeParole
##                                                 -0.57348382
##                                    Release.TypeParole Granted
##                                                 -0.42786322
##                           Release.TypeParoled to Detainer - INS
##                                                 -2.23267760
##                          Release.TypeParoled to Detainer - Iowa
##                                                  0.26539670
##                   Release.TypeParoled to Detainer - Out of State
##                                                 -0.24528007
##                 Release.TypeParoled to Detainer - U.S. Marshall
##                                                 -2.23373513
```

```
##                       Release.TypeParoled w/Immediate Discharge
##                                                     -0.72681014
##                        Release.TypeReleased to Special Sentence
##                                                     -0.08911354
##                                     Release.TypeSpecial Sentence
##                                                      0.33782978
##     Race...EthnicityAmerican Indian or Alaska Native - Hispanic
##                                                     12.17729185
## Race...EthnicityAmerican Indian or Alaska Native - Non-Hispanic
##                                                     12.74146012
##            Race...EthnicityAsian or Pacific Islander - Hispanic
##                                                     11.98709969
##        Race...EthnicityAsian or Pacific Islander - Non-Hispanic
##                                                     11.76786493
##                                Race...EthnicityBlack - Hispanic
##                                                     11.98941298
##                            Race...EthnicityBlack - Non-Hispanic
##                                                     12.34039681
##                                          Race...EthnicityWhite -
##                                                     11.47656150
##                                Race...EthnicityWhite - Hispanic
##                                                     11.96545218
##                            Race...EthnicityWhite - Non-Hispanic
##                                                     12.29106975
##                                             Age.At.Release35-44
##                                                     -0.15401094
##                                             Age.At.Release45-54
##                                                     -0.41077804
##                                       Age.At.Release55 and Older
##                                                     -0.75415637
##                                          Age.At.ReleaseUnder 25
##                                                      0.13177445
##                                                         SexMale
##                                                      0.39089015
##                     Offense.ClassificationAggravated Misdemeanor
##                                                     11.55821480
##                                   Offense.ClassificationB Felony
##                                                     11.47269561
##                                   Offense.ClassificationC Felony
##                                                     11.53417682
##                                   Offense.ClassificationD Felony
##                                                     11.35129090
##                          Offense.ClassificationFelony - Enhanced
##                                                     11.38915989
##  Offense.ClassificationFelony - Enhancement to Original Penalty
##                                                     11.57049594
##                Offense.ClassificationFelony - Mandatory Minimum
##                                                     12.01184519
##                                Offense.ClassificationOther Felony
##                                                     -0.29951690
##                         Offense.ClassificationSerious Misdemeanor
##                                                     11.55936845
##                          Offense.ClassificationSimple Misdemeanor
##                                                     -0.32954668
```

```
##                     Offense.ClassificationSpecial Sentence 2005
##                                                      12.63454834
##                                          Offense.SubtypeAnimals
##                                                     -12.56868955
##                                           Offense.SubtypeArson
##                                                      -0.80590155
##                                          Offense.SubtypeAssault
##                                                      -0.41068386
##                                         Offense.SubtypeBurglary
##                                                      -0.10423045
##                                 Offense.SubtypeDrug Possession
##                                                      -0.05818726
##                                   Offense.SubtypeFlight/Escape
##                                                       0.07091937
##                                   Offense.SubtypeForgery/Fraud
##                                                       0.07707774
##                                         Offense.SubtypeKidnap
##                                                      -2.05371654
##                               Offense.SubtypeMurder/Manslaughter
##                                                      -1.18378844
##                                    Offense.SubtypeOther Criminal
##                                                       0.02806079
##                                       Offense.SubtypeOther Drug
##                                                      -0.12655918
##                                Offense.SubtypeOther Public Order
##                                                       0.04043914
##                                    Offense.SubtypeOther Violent
##                                                      -0.75559007
##                                             Offense.SubtypeOWI
##                                                      -0.37205961
##                           Offense.SubtypeProstitution/Pimping
##                                                       0.71635572
##                                         Offense.SubtypeRobbery
##                                                      -1.02611598
##                                            Offense.SubtypeSex
##                                                      -0.74090841
##                  Offense.SubtypeSex Offender Registry/Residency
##                                                      -0.08299812
##                      Offense.SubtypeSpecial Sentence Revocation
##                                                      -0.79237817
##                                 Offense.SubtypeStolen Property
##                                                       0.84523388
##                                          Offense.SubtypeTheft
##                                                      -0.12228119
##                                       Offense.SubtypeTraffic
##                                                      -0.50881925
##                                   Offense.SubtypeTrafficking
##                                                      -0.23108988
##                                     Offense.SubtypeVandalism
##                                                      -0.34230750
##                                       Offense.SubtypeWeapons
##                                                      -0.46505990
```

Now, we can see how this model compares to our full model.

```
model.step.prob = predict(model.step, df[test,],type='response')
head(model.step.prob)
```

```
##         1         3         5         9        11        13
## 0.2289770 0.4477712 0.2119533 0.4242753 0.4493011 0.3924763
```

```
model.step.pred = rep(FALSE, nrow(df)-length(test))
model.step.pred[model.prob > 0.5] = TRUE
# confusion matrix
tbl = table(model.step.pred, df[test,]$Return.to.Prison)
tbl
```

```
##
## model.step.pred FALSE TRUE
##           FALSE  6431 2866
##           TRUE    501  605
```

```
# error rate
1-mean(model.step.pred == df[test,]$Return.to.Prison)
```

```
## [1] 0.3236566
```

```
# false positives
tbl[1, 2] / (tbl[1, 2] + tbl[2, 2])
```

```
## [1] 0.8256986
```

```
# false negatives
tbl[2, 1] / (tbl[2, 1] + tbl[1, 1])
```

```
## [1] 0.07227351
```

So, exactly the same as the full model. We can see with the following code that the coefficients are actually the same between the two models:

```
sum(coef(model.fit)==coef(model.step)) / length(coef(model.fit))
```

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

So, 100% matching coefficients. I.e., the stepwise selection found that the full model was the best.

Now, let us attempt some other models. Since we are working with exclusively categorical predictors, LDA and QDA models are inappropriate. Instead, we will test some tree ensemble models.

Tree Classifiers

First, let's build a random forest model to predict recidivism. We will find the best number of variables considered per split.

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.3
```

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

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

```r
ms = seq(1, 7)
errors = c()

# dataframe with only relevant variables
df.rel = df[, (names(df) %in% c("Return.to.Prison", "Main.Supervising.District", "Release.Type", "Race.
df.rel$Return.to.Prison = factor(df.rel$Return.to.Prison)
Y.test = df.rel[test,]$Return.to.Prison
for (i in seq(1, length(ms))) {
  m = ms[i]
  model.rf = randomForest(Return.to.Prison~.,data=df.rel, subset=train, mtry=m, importance = TRUE, na.a
  Y.rf = predict(model.rf,newdata=df.rel[test,])
  errors[i] = 1-mean(Y.rf == Y.test)
}

model.rf.df = data.frame(errors, ms)
```

We can then find which m gives us the lowest error rate.

```r
model.rf.df$m[model.rf.df$errors==min(model.rf.df$errors)]
```

```
## [1] 2
```

So, mtry=2 provides us the best results. Let's see the model for this.

```r
model.rf = randomForest(Return.to.Prison~.,data=df.rel, subset=train, mtry=2, importance = TRUE, na.acti
Y.rf = predict(model.rf,newdata=df.rel[test,])
# confusion matrix
tbl = table(Y.rf,Y.test)
tbl
```

```
##        Y.test
## Y.rf    FALSE TRUE
##   FALSE  6097 2520
##   TRUE    835  951
```

```r
# error rate
1-mean(Y.rf == Y.test)
```

```
## [1] 0.3225031
```

```r
# false positives
tbl[1, 2] / (tbl[1, 2] + tbl[2, 2])
```

```
## [1] 0.7260156
```

```r
# false negatives
tbl[2, 1] / (tbl[2, 1] + tbl[1, 1])
```

```
## [1] 0.1204559
```

We get an error rate ever so slightly lower than that of our logistic model. We have slightly better false positive rate and worse false negative rate, so our error is somewhat more balanced. Next, we can see if a boosted tree gives us further improvement. We will test a wide range of lambdas to find one optimal for our data.

Let's find which lambda was optimal.

```r
bestlambda = model.boost.df$lambda[model.boost.df$error==min(model.boost.df$error)]
bestlambda
```

```
## [1] 0.021
```

So, we have optimal lambda 0.021. Let's see the model using this parameter.

```r
model.boost = gbm(Return.to.Prison~.,data=df.rel.train,distribution="bernoulli", n.trees=1000, interacti
Y.boost.test = predict(model.boost, newdata=df.rel.test, ntrees=1000)
```

```
## Using 1000 trees...
```

```r
Y.bool = rep(FALSE, length(Y.test))
Y.bool[Y.boost.test>0]=TRUE
# confusion matrix
tbl = table(Y.bool,Y.test)
tbl
```

```
##        Y.test
## Y.bool  FALSE TRUE
##   FALSE  6296 2667
##   TRUE    636  804
```

```r
# error rate
1-mean(Y.bool==Y.test)
```

```
## [1] 0.3175046
```

```r
# false positives
tbl[1, 2] / (tbl[1, 2] + tbl[2, 2])
```

```
## [1] 0.7683665
```

```
# false negatives
tbl[2, 1] / (tbl[2, 1] + tbl[1, 1])
```

```
## [1] 0.09174841
```

So, we have our best error rate so far, though not by much. Positive and negative error rates basically in line with previous models.

Summary

So, we have the following results: Logistic model:

```
# confusion matrix
tbl.log = table(model.pred, df[test,]$Return.to.Prison)
tbl.log
```

```
##
## model.pred FALSE TRUE
##      FALSE  6431 2866
##      TRUE    501  605
```

```
# error rate
1-mean(model.pred == df[test,]$Return.to.Prison)
```

```
## [1] 0.3236566
```

```
# false positives
tbl.log[1, 2] / (tbl.log[1, 2] + tbl.log[2, 2])
```

```
## [1] 0.8256986
```

```
# false negatives
tbl.log[2, 1] / (tbl.log[2, 1] + tbl.log[1, 1])
```

```
## [1] 0.07227351
```

Random forest:

```
# confusion matrix
tbl.rf = table(Y.rf,Y.test)
tbl.rf
```

```
##        Y.test
## Y.rf    FALSE TRUE
##   FALSE  6097 2520
##   TRUE    835  951
```

```r
# error rate
1-mean(Y.rf == Y.test)
```

```
## [1] 0.3225031
```

```r
# false positives
tbl.rf[1, 2] / (tbl.rf[1, 2] + tbl.rf[2, 2])
```

```
## [1] 0.7260156
```

```r
# false negatives
tbl.rf[2, 1] / (tbl.rf[2, 1] + tbl.rf[1, 1])
```

```
## [1] 0.1204559
```

Boosted tree:

```r
# confusion matrix
tbl.boost = table(Y.bool,Y.test)
tbl.boost
```

```
##        Y.test
## Y.bool  FALSE TRUE
##   FALSE  6296 2667
##   TRUE    636  804
```

```r
# error rate
1-mean(Y.bool==Y.test)
```

```
## [1] 0.3175046
```

```r
# false positives
tbl.boost[1, 2] / (tbl.boost[1, 2] + tbl.boost[2, 2])
```

```
## [1] 0.7683665
```

```r
# false negatives
tbl.boost[2, 1] / (tbl.boost[2, 1] + tbl.boost[1, 1])
```

```
## [1] 0.09174841
```

Overall, our boosted tree model produces the best overall error rate, though our logistic model has a bit of an edge in terms of false negatives, and our random forest for false positives, though the differences in errors our not very large.

Considerations

Were we to continue, a focus would be on reduction of false positives. Though we do not have awful overall error rate, our false positives make up the vast majority of our error. For whatever reason, many non-returning prisoners are consistently classified as returning. Some issues likely stem from the fact that we do not have each class in equal proportion. It may also be worth investigating a KNN model in the future. As of now, a KNN model has difficulties due to the entirely categorical nature of the data. A thorough investigation into a robust means of calculating distance for each categorical variable could potentially be useful. But, for now, we will be satisfied with our passable boosted tree as our best model.