HW3

Business Analytics and Data Mining

Zachary Palmore

4/11/2021

Assignment 3

Purpose

In this homework assignment, we will explore, analyze and model a data set containing information on crime for various neighborhoods of a major city. Each record has a response variable indicating whether or not the crime rate is above the median crime rate (1) or not (0).

Our purpose is to build a binary logistic regression model on the training data set to predict whether the neighborhood will be at risk for high crime levels. We will provide classifications and probabilities for the evaluation data set using our binary logistic regression model. We can only use the variables given to us (or variables that are derived from the variables provided).

Introduction

```
# Packages
library(tidyverse)
library(reshape2)
library(ggpubr)
library(ggcorrplot)
library(kableExtra)
library(Amelia)
library(caret)
library(pROC)
library(psych)
library(bestNormalize)
theme_set(theme_minimal())
# Data
cdata <- read.csv("https://raw.githubusercontent.com/palmorezm/msds/main/621/HW3/crime-training-data_mo
cdata.eval <- read.csv("https://raw.githubusercontent.com/palmorezm/msds/main/621/HW3/crime-evaluation-
```

There are 466 observations in this data set with 13 different variables. Each observation is a statistical summary that indicates an attribute associated with a particular neighborhood within the major city. For example the first variable 'zn' contains the proportion of residential land zoned for lots that are over 25000 square feet. The first five observations of each variable are shown.

```
first5.tbl <- cdata[1:5,]
kbl(first5.tbl, booktabs = T, caption = "Initial Observations") %>%
  kable_styling(latex_options = c("striped", "hold_position"), full_width = T) %>%
  footnote(c("Only showing the first five observations of each variable"))
```

Table 1: Initial Observations

zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv	target
0	19.58	0	0.605	7.929	96.2	2.0459	5	403	14.7	3.70	50.0	1
0	19.58	1	0.871	5.403	100.0	1.3216	5	403	14.7	26.82	13.4	1
0	18.10	0	0.740	6.485	100.0	1.9784	24	666	20.2	18.85	15.4	1
30	4.93	0	0.428	6.393	7.8	7.0355	6	300	16.6	5.19	23.7	0
0	2.46	0	0.488	7.155	92.2	2.7006	3	193	17.8	4.82	37.9	0

Note:
Only showing the first five observations of each variable

Our target variable is labeled 'target' and it describes whether the neighborhood's crime rate is above the median or below it. Any or all of these variables could be used as our predictors except the dummy variable of 'chas' and our target variable. In the next section we will explore the relationships between these predictors and review their interaction with our target to see which of the variables make the best predictors of crime rate. A full list of variable descriptions is also provided for reference.

```
vardesc <- data.frame(matrix(c(
'zn',   'proportion of residential land zoned for large lots (over 25000 square feet)',
'indus',    'proportion of non-retail business acres per suburb',
'chas', 'a dummy var. for whether the suburb borders the Charles River (1) or not (0)',
'nox',  'nitrogen oxides concentration (parts per 10 million)',
'rm',   'average number of rooms per dwelling',
'age',  'proportion of owner-occupied units built prior to 1940',
'dis',  'weighted mean of distances to five Boston employment centers',
'rad', 'index of accessibility to radial highways',
'tax',  'full-value property-tax rate per $10,000',
'ptratio',  'pupil-teacher ratio by town',
'lstat',    'lower status of the population (percent)',
'medv', 'median value of owner-occupied homes in $1000s',
'target',   'whether the crime rate is above the median crime rate (1) or not (0)'), byrow = TRUE, ncol
colnames(vardesc) <- c('Variable', 'Description')
kbl(vardesc, booktabs = T, caption = "Variable Descriptions") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"), full_width = F)
```

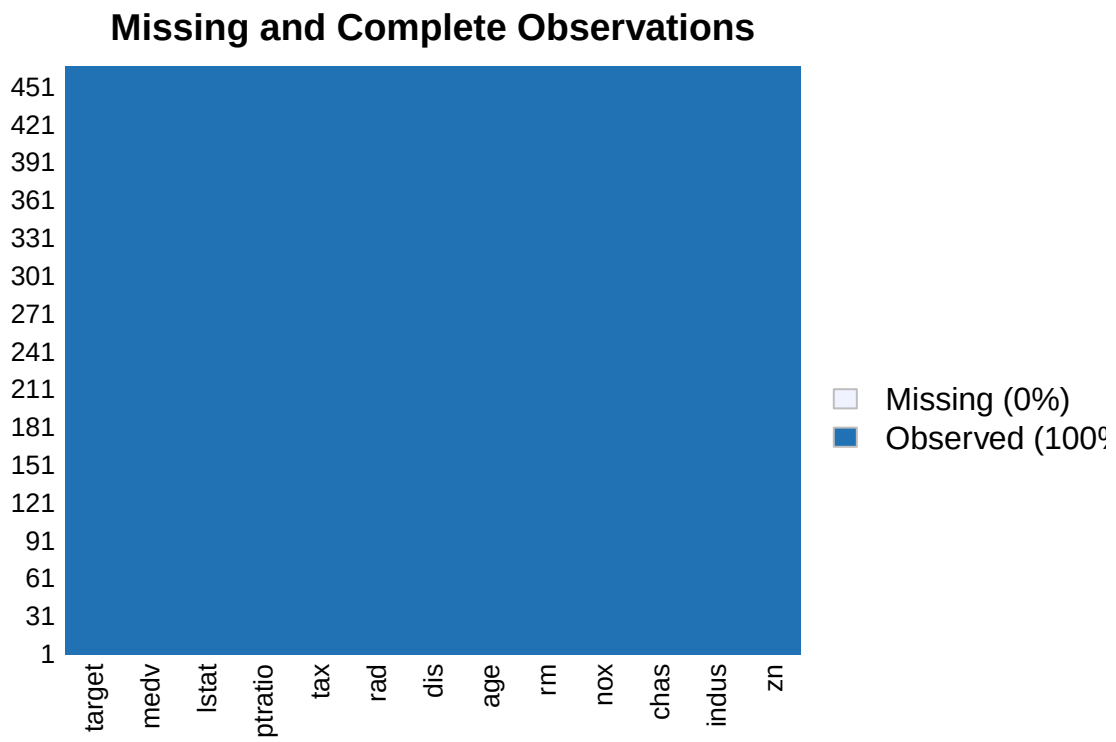
Table 2: Variable Descriptions

Variable	Description
zn	proportion of residential land zoned for large lots (over 25000 square feet)
indus	proportion of non-retail business acres per suburb
chas	a dummy var. for whether the suburb borders the Charles River (1) or not (0)
nox	nitrogen oxides concentration (parts per 10 million)
rm	average number of rooms per dwelling
age	proportion of owner-occupied units built prior to 1940
dis	weighted mean of distances to five Boston employment centers
rad	index of accessibility to radial highways
tax	full-value property-tax rate per $10,000
ptratio	pupil-teacher ratio by town
lstat	lower status of the population (percent)
medv	median value of owner-occupied homes in $1000s
target	whether the crime rate is above the median crime rate (1) or not (0)

Data Exploration

To determine which of the variables might make the best predictors we must first explore the data for potential sources of error. This includes, identifying and fixing missing values (if there are any), calculating and comparing averages, standard deviations, and other summary statistics. We begin this by checking to see if any of the data are missing. Instead of relying solely on numbers, we created a map of observations of each variable. Blue indicates the presence of an observation and light grey indicates an absence.

```
cdata.missingobservations <- sum(is.na(cdata))
missmap(cdata, main = "Missing and Complete Observations")
```



This map appears to be a solid color of blue which indicates that none of the data are missing. We can also confirm this statistically where our calculations of any non-applicable value would be counted and tallied then take the sum of those tallies for a total value missing. Perhaps as expected, the total missing from statistical calculations is 0. This is wonderful news since it means we will not need to impute but exceedingly abnormal given this kind of model.

Next we calculate and compare the averages, standard deviations, and other summary statistics. These will be used to moderate our expectations of the variables in our model and inform us of the best ways to prepare the data if there are any changes that need to be made. Those statistics are shown below.

```
summary.tbl <- describe(cdata)
desc.tbl <- summary.tbl[1:13,c(2:5, 8:10, 13)]
kbl(desc.tbl, booktabs = T, caption = "Summary Statistics - Crime Data") %>%
  kable_styling(latex_options = c("striped", "hold_position"), full_width = T)
```

Since it is quite rare to have a fully filled data set, we further confirm the presence of each observation with the 'n' column. This column describes how many observations there are of each variable. Here again, we are reassured that this data is fairly clean already with the same number of observations for each variable.

To get a sense of scale, magnitude, and direction for these vectors we calculated the mean, median, and range for each variables. Notice that the potential predictors of 'tax' and 'age' seem to be on a different scale than the remaining variables. There is also a relatively large standard error for the the variables 'zn,' 'age,' and 'tax' when compared with the other potential predictors. Many of these variables have small deviations from their means or low variance with standard deviations less than 10. However, those same three predictors mentioned, do not.

We recognize that any of these variables could confound existing relationships with our target; or may turn out to be complete outliers that could skew our model. But those three, 'zn,' 'age,' and 'tax' are enough

Table 3: Summary Statistics - Crime Data

	n	mean	sd	median	min	max	range	se
zn	466	11.5772532	23.3646511	0.00000	0.0000	100.0000	100.0000	1.0823466
indus	466	11.1050215	6.8458549	9.69000	0.4600	27.7400	27.2800	0.3171281
chas	466	0.0708155	0.2567920	0.00000	0.0000	1.0000	1.0000	0.0118957
nox	466	0.5543105	0.1166667	0.53800	0.3890	0.8710	0.4820	0.0054045
rm	466	6.2906738	0.7048513	6.21000	3.8630	8.7800	4.9170	0.0326516
age	466	68.3675966	28.3213784	77.15000	2.9000	100.0000	97.1000	1.3119625
dis	466	3.7956929	2.1069496	3.19095	1.1296	12.1265	10.9969	0.0976026
rad	466	9.5300429	8.6859272	5.00000	1.0000	24.0000	23.0000	0.4023678
tax	466	409.5021459	167.9000887	334.50000	187.0000	711.0000	524.0000	7.7778214
ptratio	466	18.3984979	2.1968447	18.90000	12.6000	22.0000	9.4000	0.1017669
lstat	466	12.6314592	7.1018907	11.35000	1.7300	37.9700	36.2400	0.3289887
medv	466	22.5892704	9.2396814	21.20000	5.0000	50.0000	45.0000	0.4280200
target	466	0.4914163	0.5004636	0.00000	0.0000	1.0000	1.0000	0.0231835

cause for concern that we need a deeper look at all predictors. We should make every attempt to avoid spurious associations if we are to build an accurate model.

Before delving into the nuances of these variables' distributions and spreads, we take a random sample of the complete table of crime data and divide it into training and evaluation data sets using a 70-30 split. Given that we had 466 observations within the complete crime data list we should expect about 327 observations in the resultant training data with the remaining 139 observations in the evaluation set.

After, we recalculate the same summary statistics on our training set called 'train.' Ideally, these identical calculations will produce values that preserve the essence of the orignial data. This helps ensure we make the best model for real-world scenarios. This process is documented below.

```
set.seed(72747)
indecies <- createDataPartition(cdata$target, p = .7, list = FALSE, times = 1)
train <- cdata[indecies,]
eval <- cdata[-indecies,]
train.summary.tbl <- train %>%
  select(-target) %>%
  describe()
t.desc.tbl <- train.summary.tbl[1:12,c(2:5, 8:10, 13)]
kbl(t.desc.tbl, booktabs = T, caption = "Summary Statistics - Training Data") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"), full_width = T)
```

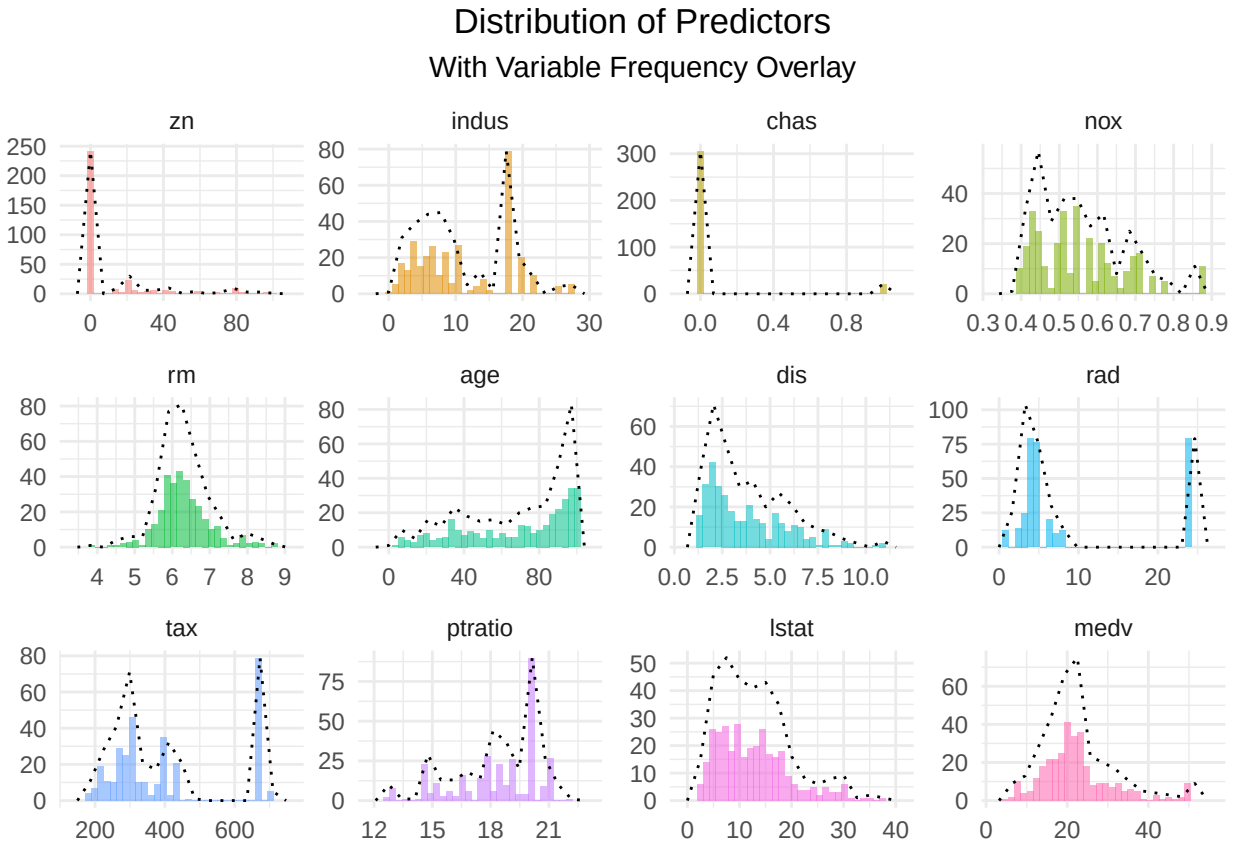
Table 4: Summary Statistics - Training Data

	n	mean	sd	median	min	max	range	se
zn	327	10.4495413	21.9233829	0.0000	0.0000	100.0000	100.0000	1.2123658
indus	327	11.1525382	6.8705743	9.6900	0.4600	27.7400	27.2800	0.3799436
chas	327	0.0642202	0.2455205	0.0000	0.0000	1.0000	1.0000	0.0135773
nox	327	0.5535804	0.1150184	0.5380	0.3920	0.8710	0.4790	0.0063605
rm	327	6.3032018	0.6969044	6.2260	3.8630	8.7250	4.8620	0.0385389
age	327	68.4428135	28.5303595	77.3000	2.9000	100.0000	97.1000	1.5777325
dis	327	3.7624006	2.0393137	3.1323	1.1742	10.7103	9.5361	0.1127743
rad	327	9.1529052	8.4983497	5.0000	1.0000	24.0000	23.0000	0.4699598
tax	327	406.2140673	165.9779996	335.0000	188.0000	711.0000	523.0000	9.1786044
ptratio	327	18.3828746	2.1993335	18.9000	12.6000	22.0000	9.4000	0.1216234
lstat	327	12.6643425	6.9518347	11.7400	1.9200	36.9800	35.0600	0.3844373
medv	327	22.3694190	8.9705883	21.0000	5.0000	50.0000	45.0000	0.4960747

Results indicate that our random sample split worked properly with 327 observations of all potential predictors in the training set. Our calculated values also remain close to the original with the most variation occurring in those same three previously mentioned variables 'zn,' 'age,', and 'tax.' Thus, the essence of the crime data tables has been persevered. These predictors will likely need adjusting but for now, we continue exploring the data.

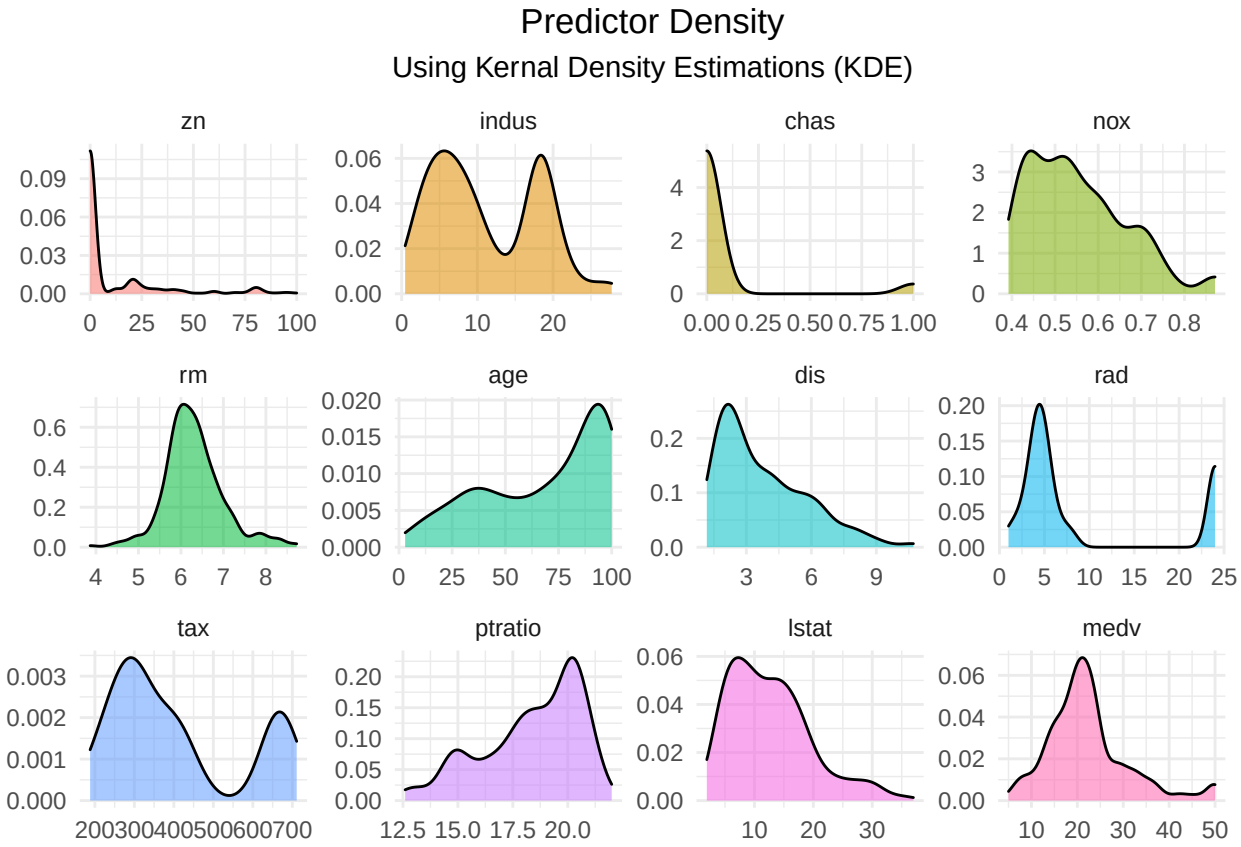
Now, we check for patterns in the distributions of the predictors. Finding those that might not fit within our model improves our precision. We can also being to check for conditional assumptions that must be made for us to be able perform a logistic regression. Keep a close look on how these variables differ in the histogram-frequency chart below.

```
train[1:12] %>%
  melt() %>%
  ggplot() +
    geom_histogram(aes(value, alpha = 2, fill = variable)) +
    facet_wrap(~ variable, scales = "free") +
    geom_freqpoly(aes(value), bins=15, lty = 3) +
    labs(title = "Distribution of Predictors", subtitle = "With Variable Frequency Overlay") +
    theme(axis.title = element_blank(),
          legend.position = "none",
          plot.title = element_text(hjust = 0.5),
          plot.title.position = "panel",
          plot.subtitle = element_text(hjust = 0.5))
```



One method to evaluate distribution is using the kernel density estimates of values present beneath each predictor. In doing so we can create another visual aid to show how the density of predictors changes over its respective distribution. This is shown below.

```
train[1:12] %>%
  melt() %>%
  ggplot() +
    geom_density(aes(value, alpha=.50, fill = variable)) +
    facet_wrap(~ variable, scales = "free") +
    labs(title = "Predictor Density",
         subtitle = "Using Kernal Density Estimations (KDE)") +
    theme(axis.title = element_blank(),
          legend.position = "none",
          plot.title = element_text(hjust = 0.5),
          plot.title.position = "panel",
          plot.subtitle = element_text(hjust = 0.5))
```



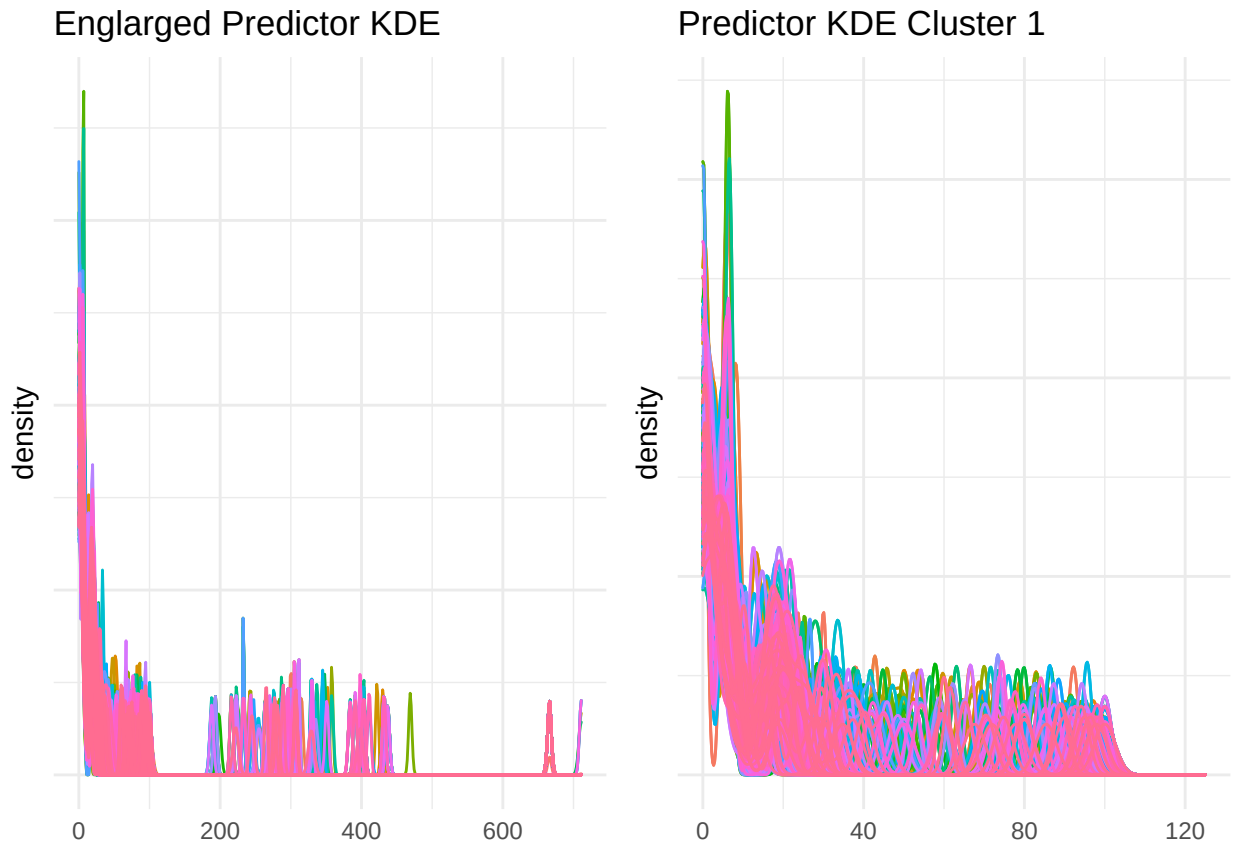
It appears some predictors are clearly being overly influenced by outliers. Meanwhile, the 'chas' distribution is binary like our target and offers no real benefit to our model. Addtionally, some predictors exhibit normality such as 'rm' and 'medv.' Unfortunately normality is not a requirement of logistic regression. Although it is an immensely useful tool to understand how the model is expected to function. In order to further improve real-world accuracy, we should remind ourselves of what we need to validate before building a model.

The conditions of binary logistic regression include: independence of observations, a dichotomous dependent variable, little to no multicollinearity, linearity of the logit odds, and a large enough sample size relative to the expected probability of the least frequent outcome. These are evaluated in several ways, the first of which we can review with a kernel density estimation with predictors plotted by color. Since there are twelve predictors, it may appear a little messy but remember we are observing overall trends. Labels will not be necessary since the name, type, or association of color with predictor is irrelevant.

```
t.train <- data.frame(t(train)) # Transpose
m.train <- train %>%
  melt()
f.train <- cbind(m.train, t.train)

# Overal kernal density estimation
kde.lim125 <- f.train %>%
  rename("predictor" = variable,
         "m.vals" = value) %>%
  melt()  %>%
  ggplot() +
  geom_density(aes(value, color = variable, alpha=.1)) +
  labs(title = "Predictor KDE Cluster 1") +
  theme(legend.position = "none",
```

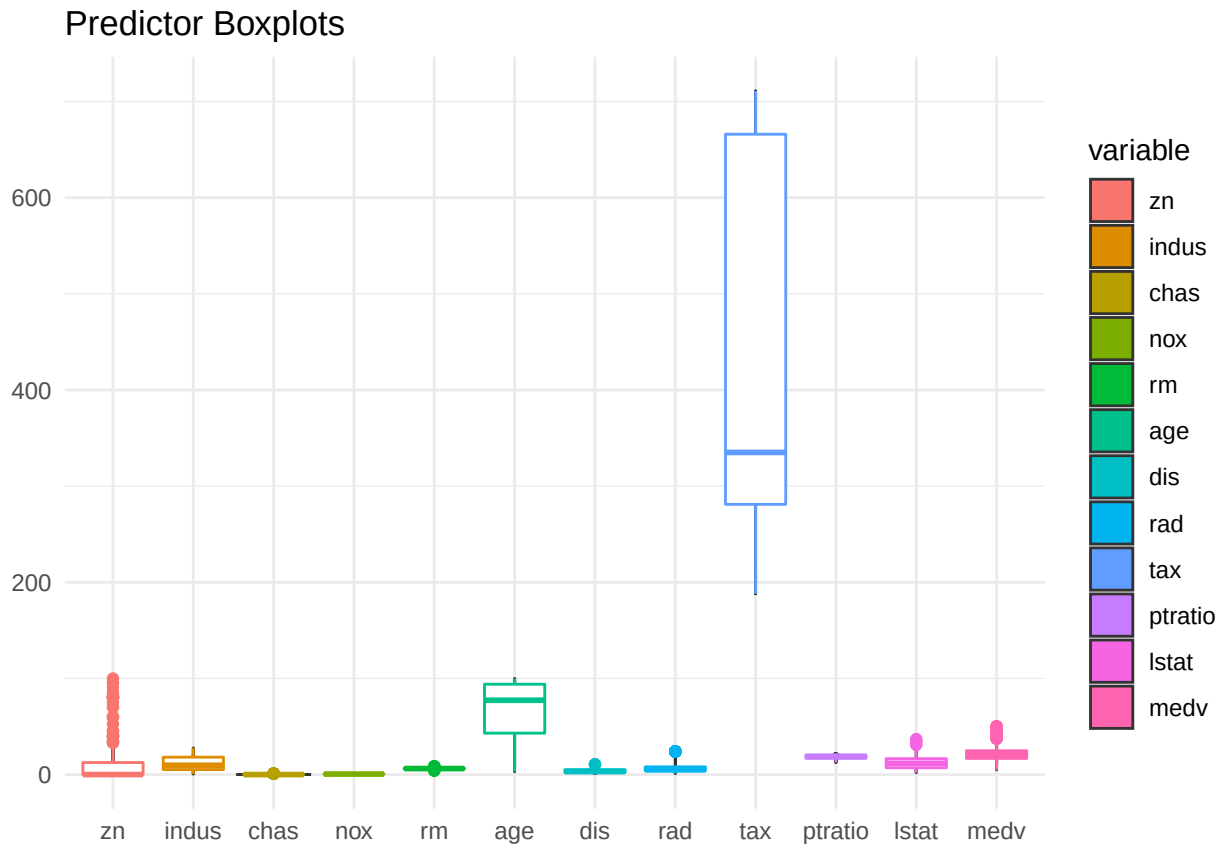
```
          axis.title.x = element_blank(),
          axis.text.y = element_blank()) +
  geom_blank(aes(value), binwidth = 5) +
  geom_density(aes(value, color = variable, alpha=.1)) +
  theme(legend.position = "none",
          axis.title.x = element_blank(),
          axis.text.y = element_blank()) +
  xlim(0, 125)
kde.nolim <- f.train %>%
  rename("predictor" = variable,
         "m.vals" = value) %>%
  melt()  %>%
  ggplot() +
  geom_density(aes(value, color = variable, alpha=.1)) +
  labs(title = "Englarged Predictor KDE") +
  theme(legend.position = "none",
          axis.title.x = element_blank(),
          axis.text.y = element_blank()) +
  geom_blank(aes(value), binwidth = 5)
ggarrange(kde.nolim, kde.lim125)
```



From this we can infer several things. First that the data is clustered into three to four major groups, in which almost all of the data is concentrated into the first group. Second, the first group contains data that appear concentrated at 0 and 1 which, in a continuous distribution of density, is not beneficial. These variables should be removed in data preparation.

This graphic also implies there are outliers that contribute to prediction by asymmetrical design and thus increased error in our model. These too should be dealt with in data preparation. Given the variation in peaks of cluster 1, it is likely that our observations are independent of one another although we cannot confirm this assumption from the graphic alone.

```
train[1:12] %>%
  melt() %>%
  ggplot(, aes(variable, value)) +
  geom_violin(aes(variable, value, fill=variable)) +
  geom_boxplot(aes(variable, value, color = variable)) +
  guides(color=FALSE, size=FALSE) +
  ggtitle("Predictor Boxplots") +
  theme(axis.title = element_blank())
```

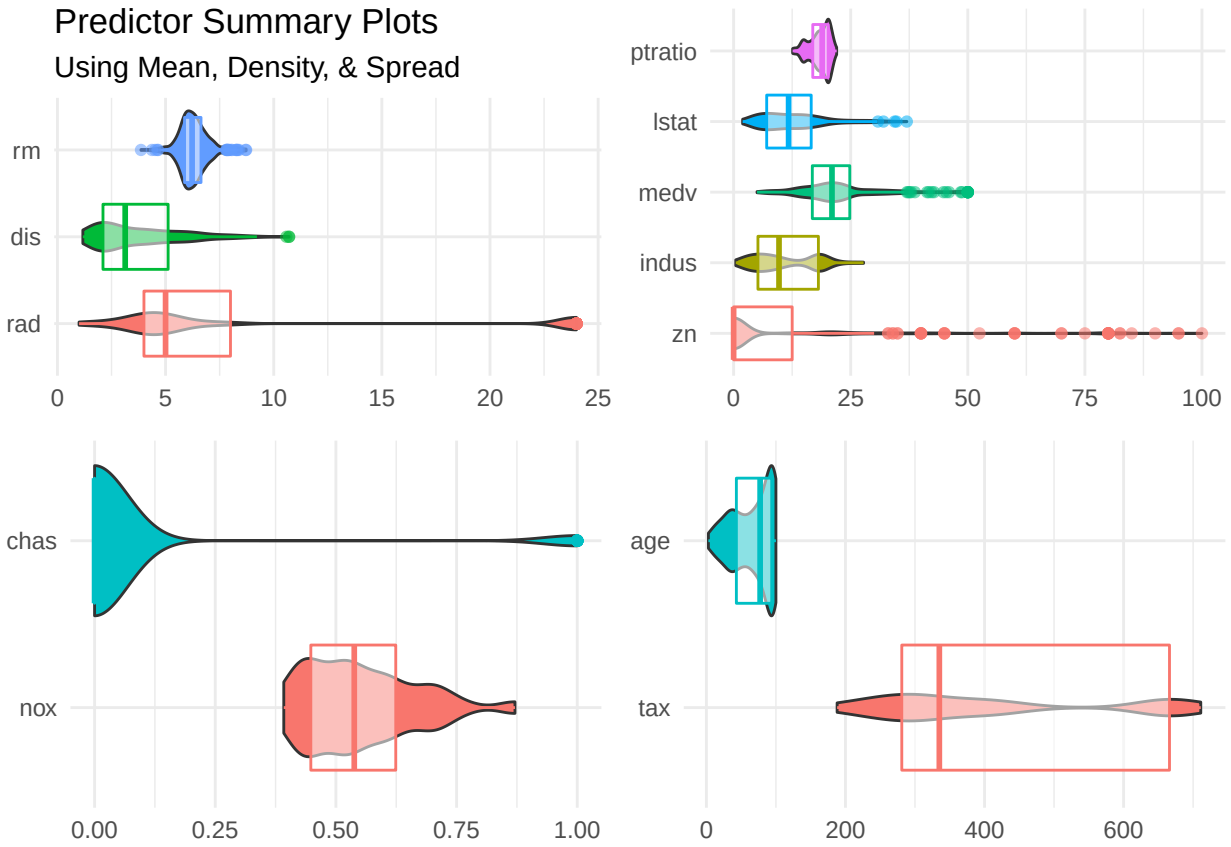


Based on the predictor boxplot above and our previous research, we can break the data into groups. In this boxplot, also note just how abnormal the predictors of 'tax' and 'age' are over the entire model. At this point, we could confidently separate them to avoid making error-prone estimates of the crime rate. This forms one group. We repeat this process over the remaining predictors until they fit with at least one other characteristically similar predictor that is based on scale, IQR, median, and outliers.

```
# seperate into groups
zimlp <- train[,c("zn", "indus", "medv", "lstat", "ptratio")]
ta <- train[,c("tax", "age")]
rdr <- train[,c("rad", "dis", "rm")]
nc <- train[,c("nox", "chas")]
```

With these predictor groups, we can create violin plots with those boxplots to describe the median and outliers of each predictor. It is another, more useful way to visualize the distribution while evaluating those conditional assumptions of logistic regression to guarantee model adheres to the rules of binary logistic modeling. Kernel density estimates are placed on here as the black lines that create the 'violin' shapes to gauge where in the distribution point are concentrated. This is shown relative to the predictor's own mean, IQR, range, deviations, and shows variation across the spread of other predictors in their clustered groups.

```
viobox.zimlp <- zimlp %>%
  melt() %>%
  ggplot() +
  geom_violin(aes(variable, value, fill=variable, fatten=2)) +
  geom_boxplot(aes(variable, value, color = variable, alpha = .90)) +
  xlab("Value") + ylab("Variable") +
  theme(legend.position = "none", axis.title = element_blank()) +
  coord_flip()
viobox.ta <- ta %>%
  melt() %>%
  ggplot() +
  geom_violin(aes(variable, value, fill=variable, fatten=2)) +
  geom_boxplot(aes(variable, value, color = variable, alpha = .90)) +
  xlab("Value") + ylab("Variable") +
  theme(legend.position = "none", axis.title = element_blank()) +
  coord_flip()
viobox.rdr <- rdr %>%
  melt() %>%
  ggplot() +
  geom_violin(aes(variable, value, fill=variable, fatten=2)) +
  geom_boxplot(aes(variable, value, color = variable, alpha = .90)) +
  xlab("Value") + ylab("Variable") + labs(title = "Predictor Summary Plots",
                                          subtitle = "Using Mean, Density, & Spread") +
  theme(legend.position = "none", axis.title = element_blank()) +
  coord_flip()
viobox.nc <- nc %>%
  melt() %>%
  ggplot() +
  geom_violin(aes(variable, value, fill=variable, fatten=2)) +
  geom_boxplot(aes(variable, value, color = variable, alpha = .90)) +
  xlab("Value") + ylab("Variable") +
  theme(legend.position = "none", axis.title = element_blank()) +
  coord_flip()
ggarrange(viobox.rdr, viobox.zimlp, viobox.nc, viobox.ta)
```

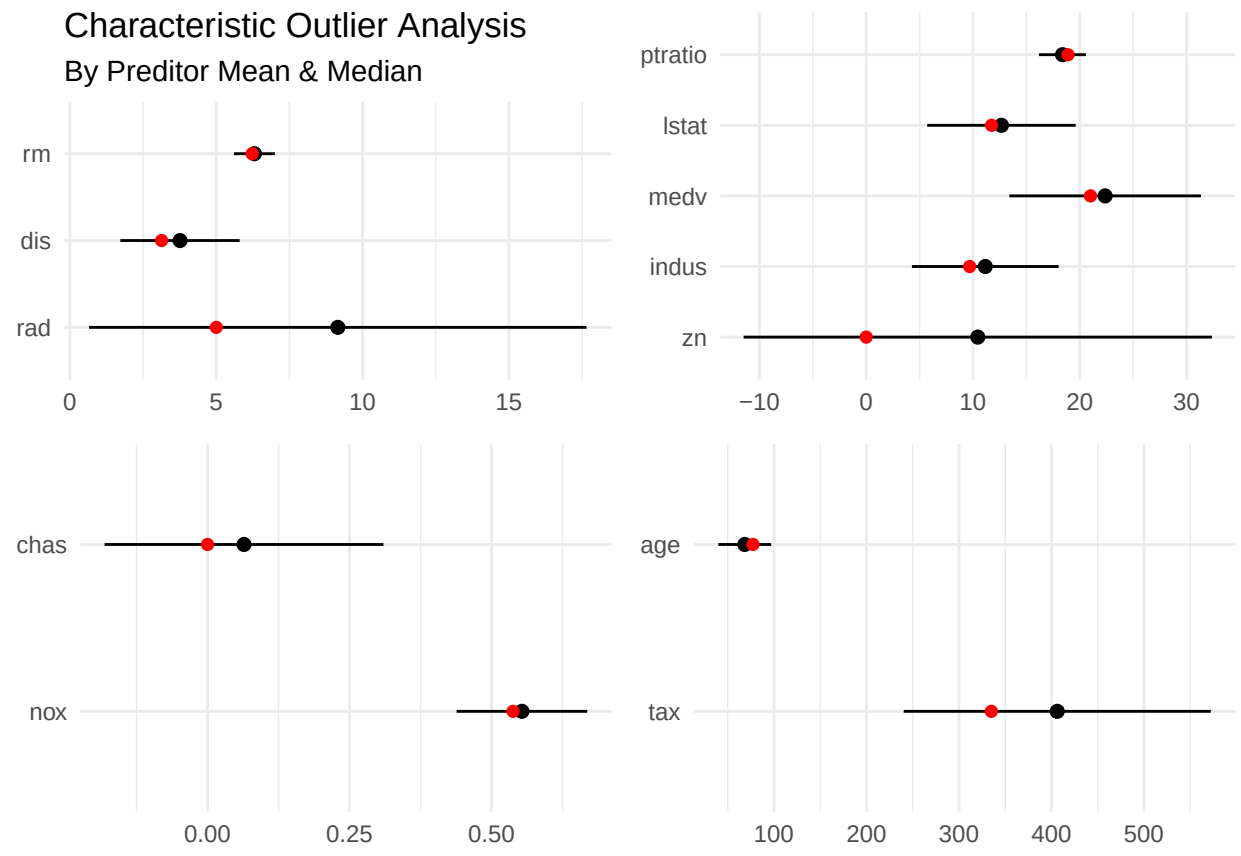


The predictors 'rm,' 'ptratio,' 'medv,' 'lstat,' and 'nox,' are the most likely choices to make an inference on at this moment. This is because they are suited to predict with; that is, they have an average near the center of their distribution, are more unimodal than other options, and contain a spread with minimal outliers. While this makes it easy to see which distributions' properties it is hard to discern exactly what the magnitude of influence outliers could play in each predictors' distribution. For that, we repeat the process with a new plot.

```
# Function to calculate and set pointrange
xysdu <- function(x) {
   m <- mean(x)
   ymin <- m - sd(x)
   ymax <- m + sd(x)
   return(c(y = m, ymin = ymin, ymax = ymax))
}

colors <- c("Median" = "Red", "Mean" = "Black")
ptrng.zimlp <- zimlp %>%
  melt() %>%
  ggplot(aes(variable, value)) + coord_flip() +
  stat_summary(fun.data=xysdu, geom = "Pointrange", shape=16, size=.5, color="black") +
  stat_summary(fun.y=median, geom="point", shape=16, size=2, color="red") +
  theme(legend.position = "None", axis.title.x = element_blank(), axis.title.y = element_blank())
ptrng.rdr <- rdr %>%
  melt() %>%
  ggplot(aes(variable, value)) + coord_flip() +
  stat_summary(fun.data=xysdu, geom = "Pointrange", shape=16, size=.5, color="black") +
```

```
  stat_summary(fun.y=median, geom="point", shape=16, size=2, color="red") +
  labs(title = "Characteristic Outlier Analysis",
       subtitle = "By Preditor Mean & Median") +
  scale_color_manual(values = colors) +
  theme(legend.position = "Bottom", axis.title.x = element_blank(), axis.title.y = element_blank())
ptrng.ta <- ta %>%
  melt() %>%
  ggplot(aes(variable, value)) + coord_flip() +
  stat_summary(fun.data=xysdu, geom = "Pointrange", shape=16, size=.5, color="black") +
  stat_summary(fun.y=median, geom="point", shape=16, size=2, color="red") +
  theme(legend.position = "None", axis.title.x = element_blank(), axis.title.y = element_blank())
ptrng.nc <- nc %>%
  melt() %>%
  ggplot(aes(variable, value)) + coord_flip() +
  stat_summary(fun.data=xysdu, geom = "Pointrange", shape=16, size=.5, color="black") +
  stat_summary(fun.y=median, geom="point", shape=16, size=2, color="red") +
  theme(legend.position = "None", axis.title.x = element_blank(), axis.title.y = element_blank())
ggarrange(ptrng.rdr, ptrng.zimlp, ptrng.nc, ptrng.ta)
```



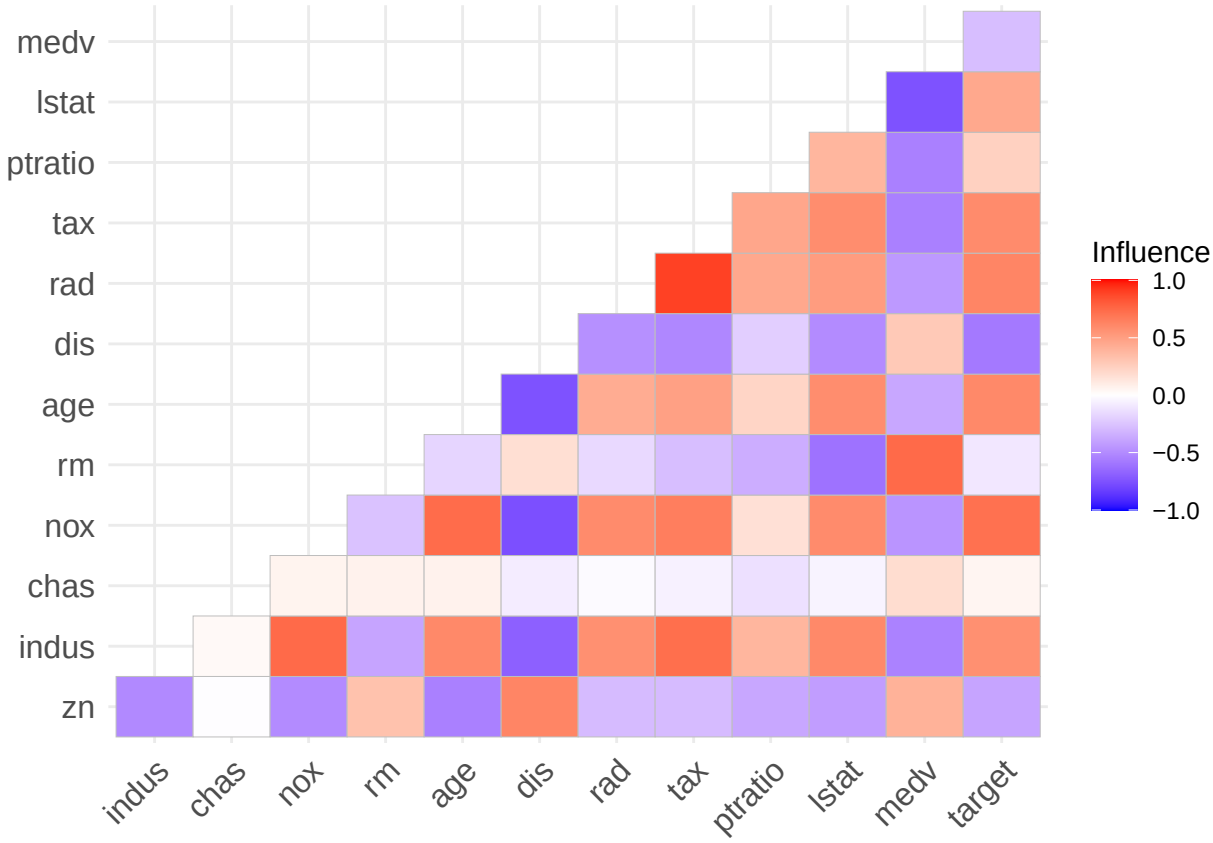
Taking advantage of a statistical phenomena, we find the pointrange of each predictor by exploiting the difference in robustness of mean and median and graphing them on top of the respective predictor ranges. Recall these predictors have already been grouped by similar values as shown in the violin plots above. These predictors are situated in the same groups and positioning for quick referencing.

When evaluating in this way, the predictors' medians are plotted as red dots on a black line that is the point range calculated from the standard deviation of each distribution. In the center of that point range is a

black dot that represents the mean of each predictor. The farther away the red and black dots are from one another, the more influence outliers have on the predictor.

This pattern emphasizes the difference within clusters to eliminate the potential for conflicts of size and scale. On a scale of similar values, we can tell just how much influence each predictor harbors compared to others in its group. In the upper left quadrant, the predictor 'rad' has the most skewness. In the upper right quadrant, 'zn.' The best predictors shown seem to be 'nox,' 'pratio,' 'lstat,' and 'rm,' with potential from 'medv,' 'age,' and 'dis.'. Although there is one more assumption we should consider, multicollinearity.

```
train %>%
  cor() %>%
  ggcorrplot(method = "square", type="upper", ggtheme = ggplot2::theme_minimal, legend.title = "Influen
```



There is a near perfect correlation in the positive direction between the predictors 'rad' and 'tax.' To avoid entanglement, these cannot be modeled simultaneously. A few other predictors, such as 'age' and 'dis' are closely related, but not close enough to exclude from the analysis.

```
# Compute estimated Sample Size
predictors <- (length(colnames(train))-1)
Pexpected <- 1 # If we expect at least one neighborhood above median
case.min <- 10 # from general rule and practice
est.samplesize <- (case.min*predictors)/Pexpected
```

To review, the dependent variable is binary which makes it dichotomous. So this condition is satisfied. We also assume based on the evidence we have, including that these continuous variables are proportions of the whole for each neighborhood, that observations made are independent of one another. Linearity of log odds

will be evaluated once the data is prepared. We could compute if the sample size is large enough through the total predictor variables and expected probability of the least frequent outcome, but the information in this data set comes from summarized proportions and averages (medians). For this reason, our best estimate of an accurate sample size for the study is about 120 if we also assume that the expected probability of our least frequent outcome is one.

Data Preparation

To prepare the data we solve the potential problems found during exploration. They were to adjust scales to ensure similar comparisons. To remove outliers and reduce the influence exerted by extreme values to maintain satisfaction with logistical modeling's conditional statements. To prevent autocorrelation and transform the data to produce a Gaussian training set that allows users to visually interpret predictions with ease. Recall, there were no missing values and thus no imputation is required.

```
# exclude nonmeaningful discrete ints, target, chas in reduction process
df <- train %>%
  select(-c(target, chas))
# remove outliers based on IQR
for (i in colnames(df)) {
  iqr <- IQR(df[[i]])
  q <- quantile(df[[i]], probs = c(0.25, 0.75), na.rm = FALSE)
  qupper <- q[2]+1.5*iqr
  qlower <- q[1]+1.5*iqr
  outlier_free <- subset(df, df[[i]] > (q[1] - 1.5*iqr) & df[[i]] < (q[2]+1.5*iqr) )
}
df <- outlier_free
```

As shown above, outliers were removed using the interquartile ranges of each predictor. We also excluded discrete integers and the target since they would not contribute meaningfully to the analysis. This reduced the dimensions of our data frame to 304 observations of 11 predictors.

In this reduction process, and where applicable, if values extended too far beyond their upper or lower ranges, they were removed preserving only those values that might be considered normal for median value or proportions of the attribute in a neighborhood. This is determined mathematically by 2 quantiles above or below the upper or lower quartiles respectively.

We continue to prepare the data with an evaluation of transformation methods. Although normalized data is not essential for categorical analysis such as this one, it is valuable for visual cues which helps us test and improve the model. In this case, we loop through each variable to determine the best method for a Gaussian display.

```
bestNorms <- df[1:11,]
for (i in colnames(df)) {
  bestNorms[[i]] <- bestNormalize(df[[i]],
                                  allow_orderNorm = FALSE,
                                  out_of_sample =FALSE)
}
```

```
bestNorms$zn$chosen_transform
```

```
## center_scale(x) Transformation with 304 nonmissing obs.
```

```
##  Estimated statistics:
##  - mean (before standardization) = 9.284539
##  - sd (before standardization) = 20.30682
```

```
bestNorms$indus$chosen_transform
```

```
## Standardized sqrt(x + a) Transformation with 304 nonmissing obs.:
##  Relevant statistics:
##  - a = 0
##  - mean (before standardization) = 3.222553
##  - sd (before standardization) = 1.061222
```

```
bestNorms$nox$chosen_transform
```

```
## Standardized Log_b(x + a) Transformation with 304 nonmissing obs.:
##  Relevant statistics:
##  - a = 0
##  - b = 10
##  - mean (before standardization) = -0.2627473
##  - sd (before standardization) = 0.08710804
```

```
bestNorms$rm$chosen_transform
```

```
## Standardized Log_b(x + a) Transformation with 304 nonmissing obs.:
##  Relevant statistics:
##  - a = 0
##  - b = 10
##  - mean (before standardization) = 0.7896718
##  - sd (before standardization) = 0.04029799
```

```
bestNorms$age$chosen_transform
```

```
## center_scale(x) Transformation with 304 nonmissing obs.
##  Estimated statistics:
##  - mean (before standardization) = 68.83684
##  - sd (before standardization) = 28.40914
```

```
bestNorms$dis$chosen_transform
```

```
## Standardized Log_b(x + a) Transformation with 304 nonmissing obs.:
##  Relevant statistics:
##  - a = 0
##  - b = 10
##  - mean (before standardization) = 0.5132231
##  - sd (before standardization) = 0.2301267
```

```
bestNorms$rad$chosen_transform
```

```
## Standardized asinh(x) Transformation with 304 nonmissing obs.:
##  Relevant statistics:
##  - mean (before standardization) = 2.568442
##  - sd (before standardization) = 0.8401209
```

```
bestNorms$tax$chosen_transform
```

```
## Standardized Box Cox Transformation with 304 nonmissing obs.:
##  Estimated statistics:
##  - lambda = -0.4566989
##  - mean (before standardization) = 2.042603
##  - sd (before standardization) = 0.02551697
```

```
bestNorms$ptratio$chosen_transform
```

```
## Standardized Box Cox Transformation with 304 nonmissing obs.:
##  Estimated statistics:
##  - lambda = 1.999958
##  - mean (before standardization) = 174.0132
##  - sd (before standardization) = 37.15375
```

```
bestNorms$lstat$chosen_transform
```

```
## Standardized Log_b(x + a) Transformation with 304 nonmissing obs.:
##  Relevant statistics:
##  - a = 0
##  - b = 10
##  - mean (before standardization) = 1.068512
##  - sd (before standardization) = 0.2254716
```

```
bestNorms$medv$chosen_transform
```

```
## Standardized Box Cox Transformation with 304 nonmissing obs.:
##  Estimated statistics:
##  - lambda = 0.8308657
##  - mean (before standardization) = 13.57581
##  - sd (before standardization) = 3.915992
```

Based on the recommendations of the bestNormalize function, there are two main transformations, Box-Cox and log_x. Rather than perform their best respective transformations on each predictor, we will only perform them on three specific variables that we suspect will predict better than most others. The process is documented in the creation of our finalized data set called 'training.'

```
# Focus on specific predictors
bxcx.medv <- boxcox(df$medv)
bxcx.ptratio <- boxcox(df$ptratio)
logx.lstat <- log_x(df$lstat)
transformed <- data.frame(logx.lstat$x, bxcx.medv$x, bxcx.ptratio$x)
transformed <- transformed %>%
  rename("lstat2"=logx.lstat.x,
         "medv2"=bxcx.medv.x,
         "ptratio2"=bxcx.ptratio.x)
evalu <- eval %>%
  select(-"chas")
zip <- cbind(df, transformed)
```

```
zip$target <- sample(train$target, 304, replace = FALSE)
zip <- zip[,c(1:8, 12:15)] %>%
  rename("lstat"=lstat2,
         "medv"=medv2,
         "ptratio"=ptratio2)
training <- zip
```

Our first model will be one based on a single predictor to build on. It is included in the preparation section to test our prepared training data and to inform us of where our model should improve when we begin building. Of course, this will include our target variable from the 'training' data set and be modeled by the 'medv' predictor; which is equivalent to median value of owner-occupied homes.

This idea considers the hypothesis that wealthier homeowners are able to buy their way out of higher crime neighborhoods or into lower crime areas. This could be through the selling of higher valued homes or alternatively excluding those who cannot afford homes that might be more likely to commit a crime. We test this here in our first model, 'mod1'.

Generally speaking, when we consider criminal activity and the rate at which it occurs, we are not speaking of fraud, money laundering, or other high crimes and misdemeanors that typically are not associated with any specific location. We are only speaking of physical, reportable crimes that often have a discrete statistic with a specific victim or victims. Meanwhile, criminal activity with an broad, non-discrete victim and no physical location remain unreported. It is not something we can adjust for in this analysis, but it is worth discussing.

```
mod1 <- glm(target ~ medv, family = binomial(link = "logit"), training)
summary(mod1)
```

```
##
## Call:
## glm(formula = target ~ medv, family = binomial(link = "logit"),
##     data = training)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.147  -1.139  -1.131   1.217   1.226
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.064498   0.383451  -0.168    0.866
## medv        -0.001341   0.017727  -0.076    0.940
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.79  on 303  degrees of freedom
## Residual deviance: 420.78  on 302  degrees of freedom
## AIC: 424.78
##
## Number of Fisher Scoring iterations: 3
```

Our accuracy, precision, and other evaluation metrics can be calculated with a confusion matrix. We create this matrix by comparing our predicted values to the evaluation data set captured in the introduction. Importantly, we devised a function to compute F1 scores which will serve as the most useful guiding statistic in prediction evaluation. Results of this first model are shown below:

```r
pred.1 <- ifelse(predict.glm(mod1, evalu,"response") >= 0.5,1,0)
cm <- confusionMatrix(factor(pred.1),
                      factor(evalu$target),"1")
results <- tibble(model = "Model 1",
                  predictors = 1,
                  F1 = cm$byClass[7],
                  deviance=mod1$deviance,
                  r2 = 1 - mod1$deviance/mod1$null.deviance,
                  aic=mod1$aic)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 64 75
##          1  0  0
##
##                Accuracy : 0.4604
##                  95% CI : (0.3756, 0.547)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : 0.9746
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.4604
##              Prevalence : 0.5396
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 1
##
```

```r
# F1 score Function
F1 <- function(c) {
  sensitivity <- c$byClass[[1]]
  precision <- c$byClass[[5]]
  return( (2*precision*sensitivity) /
            (precision + sensitivity))
}
F1.mod1 <- F1(cm)
```

This model's F1 score is NA with an overall accuracy of about 0.46. This was expected. With the only predictor being medv and the transformation and reduction of this variable, this result is intentionally low. The goal is to be able to improve and this does exactly that. Now that we have the data prepared and the process tested, we can begin building other models to beat our prepared single predictor model.

Modeling Building

When building these it is imparative that we test the data so that we can comprehend how the model functioned with its features. With this analysis, we will place a focus on the F1 score calculation in testing for later evaluation during model selection. We begin by attempting to beat the scores of our prepared model. In this case, it means trying to include the most useful predictors that may increase our accuracy. This should not be too difficult since it was our intent to start at a low value. The same process documented in our data preparation model is utilized.

```
mod1.all <- glm(target ~ ., family = binomial(link = "logit"), training)
summary(mod1.all)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = "logit"),
##     data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6771  -1.1162  -0.8915   1.1833   1.5475
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.3867674  3.2554029  -1.962  0.04977 *
## zn          -0.0088079  0.0086350  -1.020  0.30772
## indus        0.0115716  0.0332987   0.348  0.72821
## nox          2.1447969  2.1121625   1.015  0.30989
## rm           0.4005273  0.2984482   1.342  0.17959
## age         -0.0008903  0.0072152  -0.123  0.90180
## dis          0.3196756  0.1211196   2.639  0.00831 **
## rad         -0.0015497  0.0329525  -0.047  0.96249
## tax          0.0002570  0.0020040   0.128  0.89794
## lstat        0.0509625  0.0332296   1.534  0.12512
## medv         0.0222571  0.0375966   0.592  0.55385
## ptratio      0.0107552  0.0759784   0.142  0.88743
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 420.79  on 303  degrees of freedom
## Residual deviance: 410.01  on 292  degrees of freedom
## AIC: 434.01
##
## Number of Fisher Scoring iterations: 4
```

```
pred.2 <- ifelse(predict.glm(mod1.all, evalu,"response") >= 0.5,1,0)
cm <- confusionMatrix(factor(pred.2),
                      factor(evalu$target),"1")
results <- rbind(results, tibble(model = "Model 1 All",
                 predictors = 11,
                 F1 = cm$byClass[7],
                 deviance=mod1.all$deviance,
```

```
                  r2 = 1 - mod1.all$deviance/mod1.all$null.deviance,
                  aic=mod1.all$aic))
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 31 48
##          1 33 27
##
##                Accuracy : 0.4173
##                  95% CI : (0.3343, 0.5039)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : 0.9985
##
##                   Kappa : -0.153
##
##  Mcnemar's Test P-Value : 0.1198
##
##             Sensitivity : 0.3600
##             Specificity : 0.4844
##          Pos Pred Value : 0.4500
##          Neg Pred Value : 0.3924
##              Prevalence : 0.5396
##          Detection Rate : 0.1942
##    Detection Prevalence : 0.4317
##       Balanced Accuracy : 0.4222
##
##        'Positive' Class : 1
##
```

```
F1.mod1.all <- F1(cm)
```

Including all predictor variables improved the model performance moderately from the first model maintaining our accuracy and raising the model's F1 score to 0.4. However, this is still far from useful at prediction. At the moment we're not much better than random chance. To solve this, we review how the model would perform with minor alteration. We will call this 'mod2' for our second model using the second training data frame. Here again, we repeat the process with the same singular predictor selected previously.

```
mod2 <- glm(target ~ medv, family = binomial(link = "logit"), train)
summary(mod2)
```

```
##
## Call:
## glm(formula = target ~ medv, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6082  -1.1061  -0.7153   1.0740   2.1290
##
```

```
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.48174    0.34731   4.266 1.99e-05 ***
## medv        -0.07277    0.01527  -4.764 1.90e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 452.21  on 326  degrees of freedom
## Residual deviance: 424.58  on 325  degrees of freedom
## AIC: 428.58
##
## Number of Fisher Scoring iterations: 4
```

```
pred.3 <- ifelse(predict.glm(mod2, eval,"response") >= 0.5,1,0)
cm <- confusionMatrix(factor(pred.3),
                      factor(eval$target),"1")
results <- rbind(results, tibble(model = "Model 2",
                 predictors = 1,
                 F1 = cm$byClass[7],
                 deviance=mod2$deviance,
                 r2 = 1 - mod2$deviance/mod2$null.deviance,
                 aic=mod2$aic))
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 50 29
##          1 14 46
##
##                Accuracy : 0.6906
##                  95% CI : (0.6067, 0.7662)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : 0.0002004
##
##                   Kappa : 0.3879
##
##  Mcnemar's Test P-Value : 0.0327626
##
##             Sensitivity : 0.6133
##             Specificity : 0.7812
##          Pos Pred Value : 0.7667
##          Neg Pred Value : 0.6329
##              Prevalence : 0.5396
##          Detection Rate : 0.3309
##    Detection Prevalence : 0.4317
##       Balanced Accuracy : 0.6973
##
##        'Positive' Class : 1
##
```

```
F1.mod2 <- F1(cm)
```

This model with minor changes and only a single predictor performs better than the full scale model with all predictors of the training set. We had an accuracy of approximately 0.69 and and F1 score of 0.6814815. This is an improvement and so we move forward with this prepared data. Next we include all predictors for model 2. Recall that, this data set includes everything from the original crime data set which should give a better picture of what is happening with crime rate.

```
mod2.all <- glm(target ~ ., family = binomial(link = "logit"), train)
summary(mod2.all)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7335  -0.2882  -0.0071   0.0057   3.3766
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -40.230946   7.363260  -5.464 4.66e-08 ***
## zn           -0.053668   0.040564  -1.323  0.18582
## indus        -0.035706   0.053396  -0.669  0.50368
## chas          0.663377   0.849869   0.781  0.43506
## nox          44.616676   8.704457   5.126 2.96e-07 ***
## rm           -0.406722   0.790772  -0.514  0.60702
## age           0.036620   0.015699   2.333  0.01966 *
## dis           0.813324   0.257703   3.156  0.00160 **
## rad           0.586141   0.188001   3.118  0.00182 **
## tax          -0.005606   0.003092  -1.813  0.06982 .
## ptratio       0.383478   0.139129   2.756  0.00585 **
## lstat         0.063439   0.063522   0.999  0.31795
## medv          0.186034   0.078462   2.371  0.01774 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 452.21  on 326  degrees of freedom
## Residual deviance: 148.09  on 314  degrees of freedom
## AIC: 174.09
##
## Number of Fisher Scoring iterations: 9
```

```
pred.4 <- ifelse(predict.glm(mod2.all, eval,"response") >= 0.5,1,0)
cm <- confusionMatrix(factor(pred.4),
                      factor(eval$target),"1")
results <- rbind(results, tibble(model = "Model 2 All",
                  predictors = 12,
                  F1 = cm$byClass[7],
```

```
                  deviance=mod2.all$deviance,
                  r2 = 1 - mod2.all$deviance/mod2.all$null.deviance,
                  aic=mod2.all$aic))
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 62  7
##          1  2 68
##
##                Accuracy : 0.9353
##                  95% CI : (0.8806, 0.97)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8704
##
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 0.9067
##             Specificity : 0.9688
##          Pos Pred Value : 0.9714
##          Neg Pred Value : 0.8986
##              Prevalence : 0.5396
##          Detection Rate : 0.4892
##    Detection Prevalence : 0.5036
##       Balanced Accuracy : 0.9377
##
##        'Positive' Class : 1
##
```

```
F1.mod2.all <- F1(cm)
```

This is our best model yet. Our accuracy has shot upwards to about 0.94 with an F1 score of 0.937931. However, there are still a few unexpected hickups. Several of these predictors are not significant predictors of our target crime rate. This combination of predictors also has enough false positive errors that we may be able to decrease our number of predictors to form another exquisite model with higher accuracy and precision yet again. To identify which predictors should be taken out of our final model we employ a backwards and forwards selection process to impliment only those predictors that improve our score when added. This results in our third model, 'mod3' by pulling the best parts of aforementioned models while eliminating the worst in an attempt to make a better prediction.

```
# eliminate variables as they improve accuracy and F1 score
mod3 <- glm(target ~ . -chas -rm, family = binomial(link = "logit"), train)
summary(mod3)
```

```
##
## Call:
## glm(formula = target ~ . - chas - rm, family = binomial(link = "logit"),
##     data = train)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7704  -0.3089  -0.0062   0.0061   3.3640
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -39.857566   7.137682  -5.584 2.35e-08 ***
## zn           -0.060746   0.040447  -1.502 0.133131
## indus        -0.025445   0.051736  -0.492 0.622847
## nox          42.425965   8.225378   5.158 2.50e-07 ***
## age           0.033728   0.013761   2.451 0.014244 *
## dis           0.766689   0.245637   3.121 0.001801 **
## rad           0.603337   0.179209   3.367 0.000761 ***
## tax          -0.006164   0.003024  -2.038 0.041547 *
## ptratio       0.338859   0.124523   2.721 0.006503 **
## lstat         0.078748   0.059625   1.321 0.186596
## medv          0.157899   0.050049   3.155 0.001606 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 452.21  on 326  degrees of freedom
## Residual deviance: 149.01  on 316  degrees of freedom
## AIC: 171.01
##
## Number of Fisher Scoring iterations: 9
```

```
pred.5 <- ifelse(predict.glm(mod3, eval,"response") >= 0.5,1,0)
cm <- confusionMatrix(factor(pred.5),
                      factor(eval$target),"1")
results <- rbind(results, tibble(model = "Model 3",
                 predictors = 9,
                 F1 = cm$byClass[7],
                 deviance=mod3$deviance,
                 r2 = 1 - mod3$deviance/mod3$null.deviance,
                 aic=mod3$aic))
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 62  6
##          1  2 69
##
##                Accuracy : 0.9424
##                  95% CI : (0.8897, 0.9748)
##     No Information Rate : 0.5396
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8847
##
```

```
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.9200
##             Specificity : 0.9688
##          Pos Pred Value : 0.9718
##          Neg Pred Value : 0.9118
##              Prevalence : 0.5396
##          Detection Rate : 0.4964
##    Detection Prevalence : 0.5108
##       Balanced Accuracy : 0.9444
##
##        'Positive' Class : 1
##
```

```
F1.mod3 <- F1(cm)
```

Through this forward selection of predictors based on insignificance, backward and forward selection of the remaining signifcant values with a p-value greater than 0.1 we determined that the only predictors to control were 'chas' and 'rm'. Excluding 'chas' marginally increased model accuracy and F1 score while the exclusion of 'rm' exhibited no changes. For this reason the predictor 'rm' was ultimately exlcuded because it did not add value to the model. Our final accuracy measurement was 0.9424 with an F1 score of 0.9452055.

Model Selection

Our criteria for selecting the best binary logistic regression model relies on predominantly on model accuracy, especially through its F1 score. This is based on the theory that in the real-world application of these models, the amount of false positives and false negatives in our prediction are critical for analysis. The F score also combines information from its formula on model sensitivity and precision. We also consider the number of predictors, with a preference for a lower number. Thinking in terms of applications of data collection, it might be more cost-effective if we did not need to collect as many predictors but were able to make the same predictions with minimal disturbance of our accuracy. As added measures, the coefficient of determination, deviance, and Akaike information criterion (AIC) are computed for each model. Should existing accuracy and predictors lead to narrow selection margin, a higher prediction accuracy in the form of a coefficient of determination, lower deviance, and better quality model as determined by AIC will be the ultimate decider. We arrange these statistics into a table:

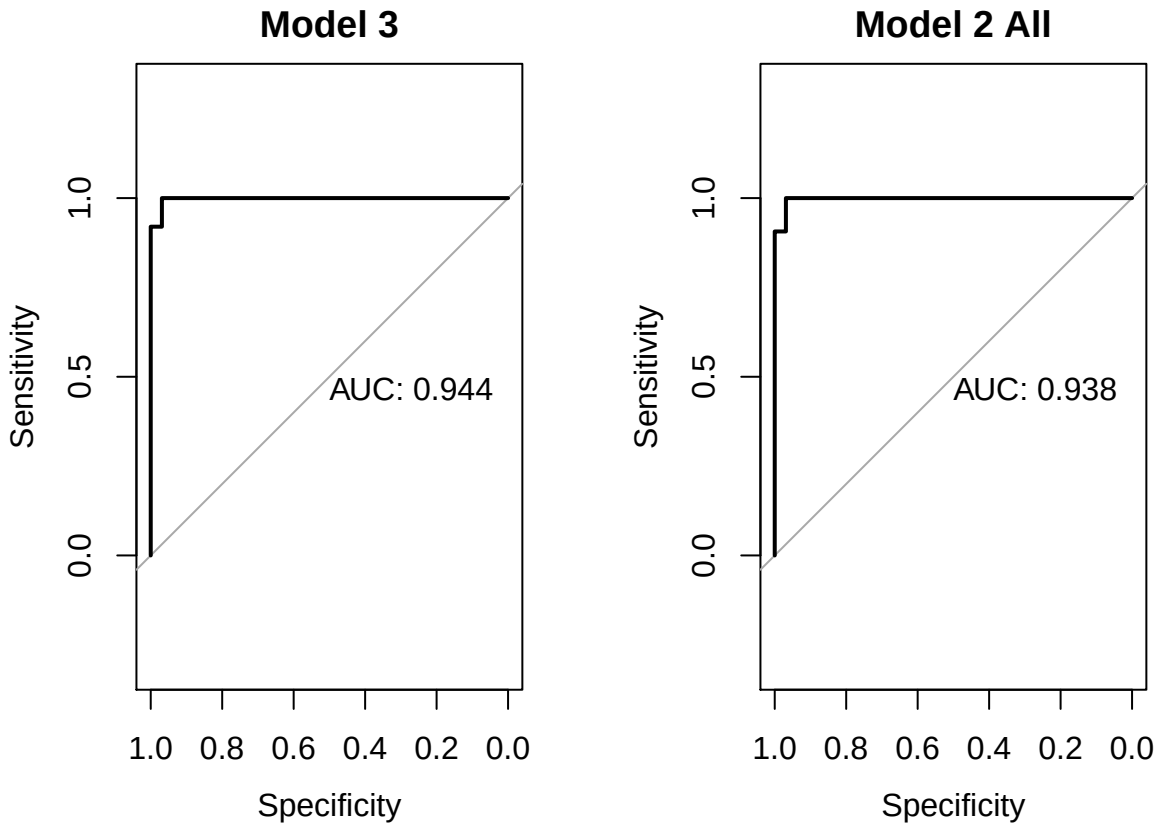
```
res.tbl <- results
kbl(res.tbl, booktabs = T, caption = "Model Results") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"), full_width = T)
```

Table 5: Model Results

model	predictors	F1	deviance	r2	aic
Model 1	1	NA	420.7828	0.0000136	424.7828
Model 1 All	11	0.4000000	410.0116	0.0256113	434.0116
Model 2	1	0.6814815	424.5783	0.0611112	428.5783
Model 2 All	12	0.9379310	148.0927	0.6725162	174.0927
Model 3	9	0.9452055	149.0135	0.6704800	171.0135

Model 2 with all predictors and Model 3 contain the best statistics. Final selection will be between those two. Keep in mind, the first model, 'mod1' was intentionally made to produce poor results. If for some reason it had turned out to be one of the best predictors then we would know something was wrong. Thankfully, this is not the case. With slightly better F1 score, lower AIC, and lower deviance as well as fewer predictors required to make the prediction than the rest of the models, Model 3, appears to be the best selection possible. However, we will evaluate through their sensitivity and specificity in a receiver operating characteristic (ROC) plot with area under the curve (AUC) displayed for these two models.

```
par(mfrow=c(1,2))
pROC::plot.roc(eval$target, pred.5, print.auc=TRUE, type="s", main="Model 3")
pROC::plot.roc(eval$target, pred.4, print.auc=TRUE, type="s", main="Model 2 All")
```



As anticipated, the Model 3 performed better in this ROC plot and has a slightly better AUC. Due to its very improved sensitivity this model contains the best performance numbers for us to go with. In addition, it uses fewer predictors to reach that level of accuracy, sensitivity, precision, and specificity. This model is the best of the available models we could select from in this analysis.

Conclusion

Due to its improved performance statistics and use of fewer predictors to achieve the same or better levels of accuracy than other models, Model 3 is the best choice for making predictions with in a real-world setting. This model has minimal changes made to it, with minor adjustments in outlier quantity a subsequent reduction of overall observations and focusing of predictors on scale by clustering. If this analysis were to

go further, it might be useful to identify other model types that could improve prediction accuracy while decreasing the predictors needed to make correct predictions. This might include an ensemble model that utilizes new predictors formed from the proportions of original predictors. However, for our purposes, an F1 score of 0.9452055 and accuracy of about 94.4% using only 9 raw predictors, will suffice for practical predictions.