# homework3

Alice Ding

2023-10-23

## Overview

In this homework assignment, you 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).

Your objective 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. You will provide classifications and probabilities for the evaluation data set using your binary logistic regression model. You can only use the variables given to you (or variables that you derive from the variables provided). Below is a short description of the variables of interest in the data set:

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

## Data Exploration

First, we'll view the summary and then we'll check if there are data points missing. Then, we'll clean the fields up to make sure they're ready for analysis.

```r
training <- read.csv('https://raw.githubusercontent.com/addsding/data621/main/homework3/crime-training-d
evaluation <- read.csv('https://raw.githubusercontent.com/addsding/data621/main/homework3/crime-evaluati

summary <- as.data.frame(describe(training))
nulls <- 466 - summary['n']
nulls_pct <- nulls / 446
summary['nulls'] <- nulls
summary['nulls_pct'] <- nulls_pct
kable(summary, digits=2) |>
```

|  | vars | n | mean | sd | median | trimmed | mad | min | max | range | skew | kurtosis | se |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| zn | 1 | 466 | 11.58 | 23.36 | 0.00 | 5.35 | 0.00 | 0.00 | 100.00 | 100.00 | 2.18 | 3.81 | 1.08 |
| indus | 2 | 466 | 11.11 | 6.85 | 9.69 | 10.91 | 9.34 | 0.46 | 27.74 | 27.28 | 0.29 | -1.24 | 0.32 |
| chas | 3 | 466 | 0.07 | 0.26 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 1.00 | 3.34 | 9.15 | 0.01 |
| nox | 4 | 466 | 0.55 | 0.12 | 0.54 | 0.54 | 0.13 | 0.39 | 0.87 | 0.48 | 0.75 | -0.04 | 0.01 |
| rm | 5 | 466 | 6.29 | 0.70 | 6.21 | 6.26 | 0.52 | 3.86 | 8.78 | 4.92 | 0.48 | 1.54 | 0.03 |
| age | 6 | 466 | 68.37 | 28.32 | 77.15 | 70.96 | 30.02 | 2.90 | 100.00 | 97.10 | -0.58 | -1.01 | 1.31 |
| dis | 7 | 466 | 3.80 | 2.11 | 3.19 | 3.54 | 1.91 | 1.13 | 12.13 | 11.00 | 1.00 | 0.47 | 0.10 |
| rad | 8 | 466 | 9.53 | 8.69 | 5.00 | 8.70 | 1.48 | 1.00 | 24.00 | 23.00 | 1.01 | -0.86 | 0.40 |
| tax | 9 | 466 | 409.50 | 167.90 | 334.50 | 401.51 | 104.52 | 187.00 | 711.00 | 524.00 | 0.66 | -1.15 | 7.78 |
| ptratio | 10 | 466 | 18.40 | 2.20 | 18.90 | 18.60 | 1.93 | 12.60 | 22.00 | 9.40 | -0.75 | -0.40 | 0.10 |
| lstat | 11 | 466 | 12.63 | 7.10 | 11.35 | 11.88 | 7.07 | 1.73 | 37.97 | 36.24 | 0.91 | 0.50 | 0.33 |
| medv | 12 | 466 | 22.59 | 9.24 | 21.20 | 21.63 | 6.00 | 5.00 | 50.00 | 45.00 | 1.08 | 1.37 | 0.43 |
| target | 13 | 466 | 0.49 | 0.50 | 0.00 | 0.49 | 0.00 | 0.00 | 1.00 | 1.00 | 0.03 | -2.00 | 0.02 |

```r
kable_styling(c("striped", "scale_down")) |>
scroll_box(width = "100%")
```

It looks like there are no nulls in the data to start which is good as this means there is no need to impute any nulls.

It appears we also have a few highly skewed variables due to many medians being quite different from the means. Some examples include the variables `zn` and `tax`.

### Class Bias Check

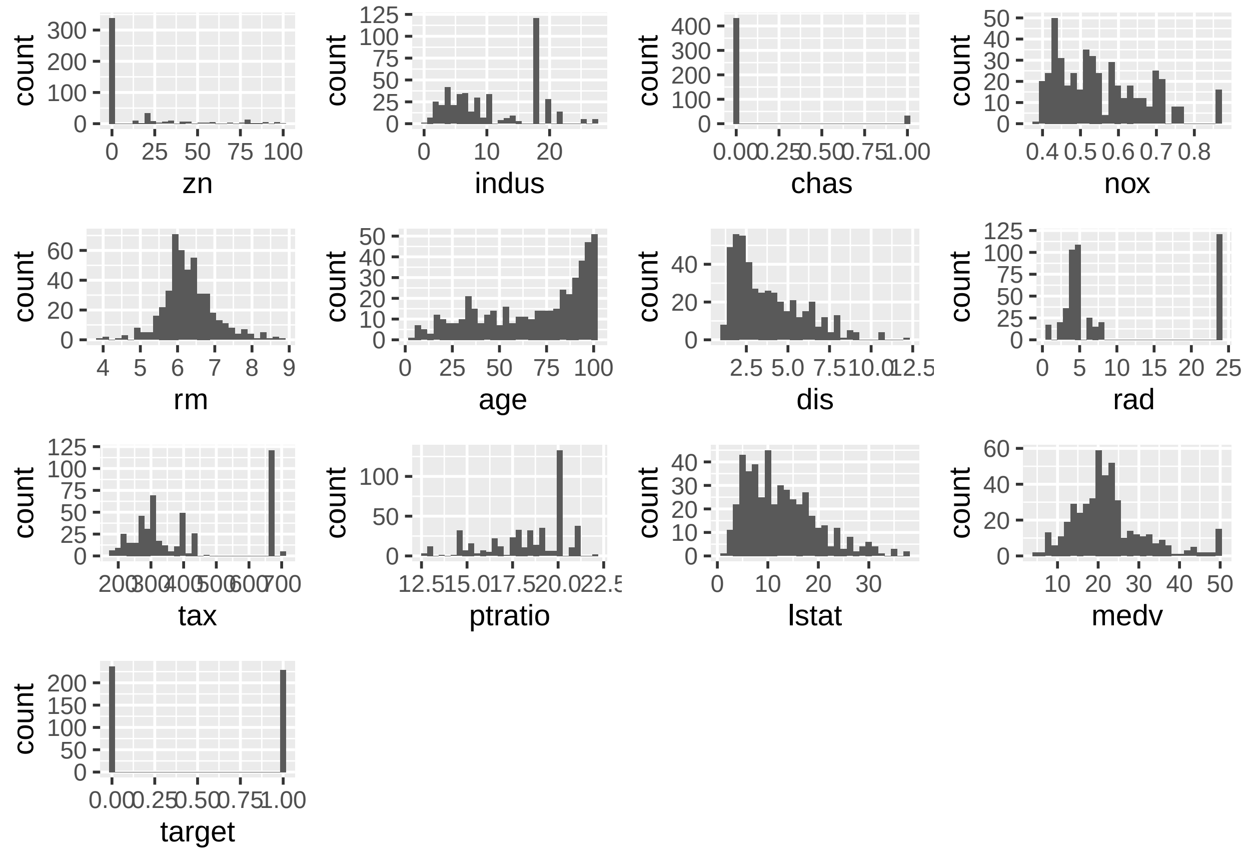
Given we only have two target values, 0 and 1, we ideally want an equal representation of both. If class imbalance were to deviate, our model performance would suffer both from effects of differential variance between the classes and bias, thus picking the more represented class. For logistic regression, if we see a strong imbalance, we can:

- up-sample the smaller group (e.g. bootstrapping),
- down-sample the larger group (e.g. sampling or bootstrapping)
- adjust our threshold for assigning the predicted value away from 0.5.

Given our target variable though seems to be relatively balanced (average of 0.49, meaning it's very close to 50% 0 and 50% 1), no upsampling or downsampling will be required to achieve class balance with this dataset.

### Distributions

Let's see what all of these fields look like distribution wise.

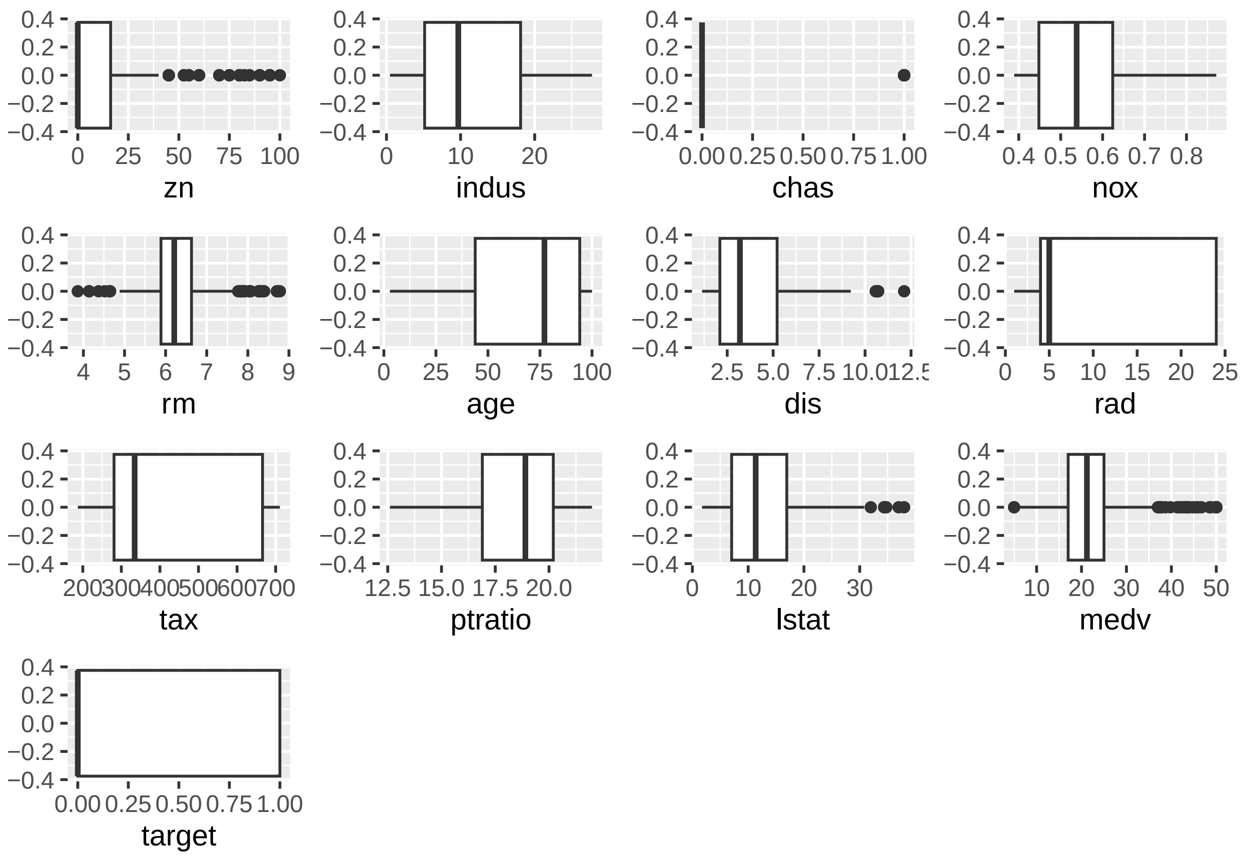


At first glance, it looks like these fields are relatively normal or have a good curve:

- `rm`
- `medv`

The rest either are pretty skewed in either direction or have no pattern really at all. Like we said in the previous section, `zn` and `tax` are skewed, however for the latter it actually looks like there's just a large amount of outliers while the beginning of the data (left side) is actually relatively normal in distribution interestingly enough.

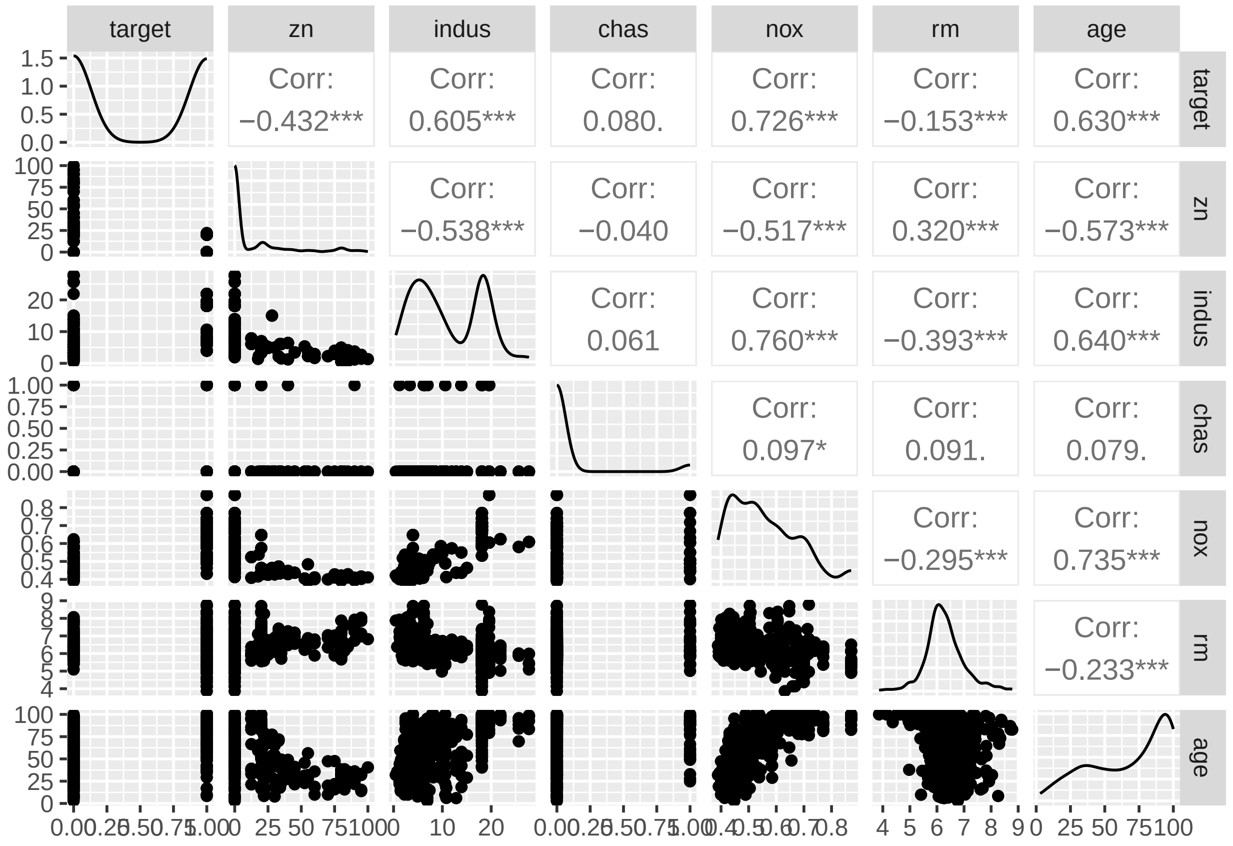
How do these look as boxplots?



In many of these fields, there seems to be quite a lot of outliers that may need to be imputed such as `zn`, `rm`, `dis`, `lstat`, and `medv`.

Now that we have a sense of how the data is distributed, what do the relationships between the variables as well as with our target look like?

Let's see how each of these fields correlates with `target` – we'll start with the first six fields.

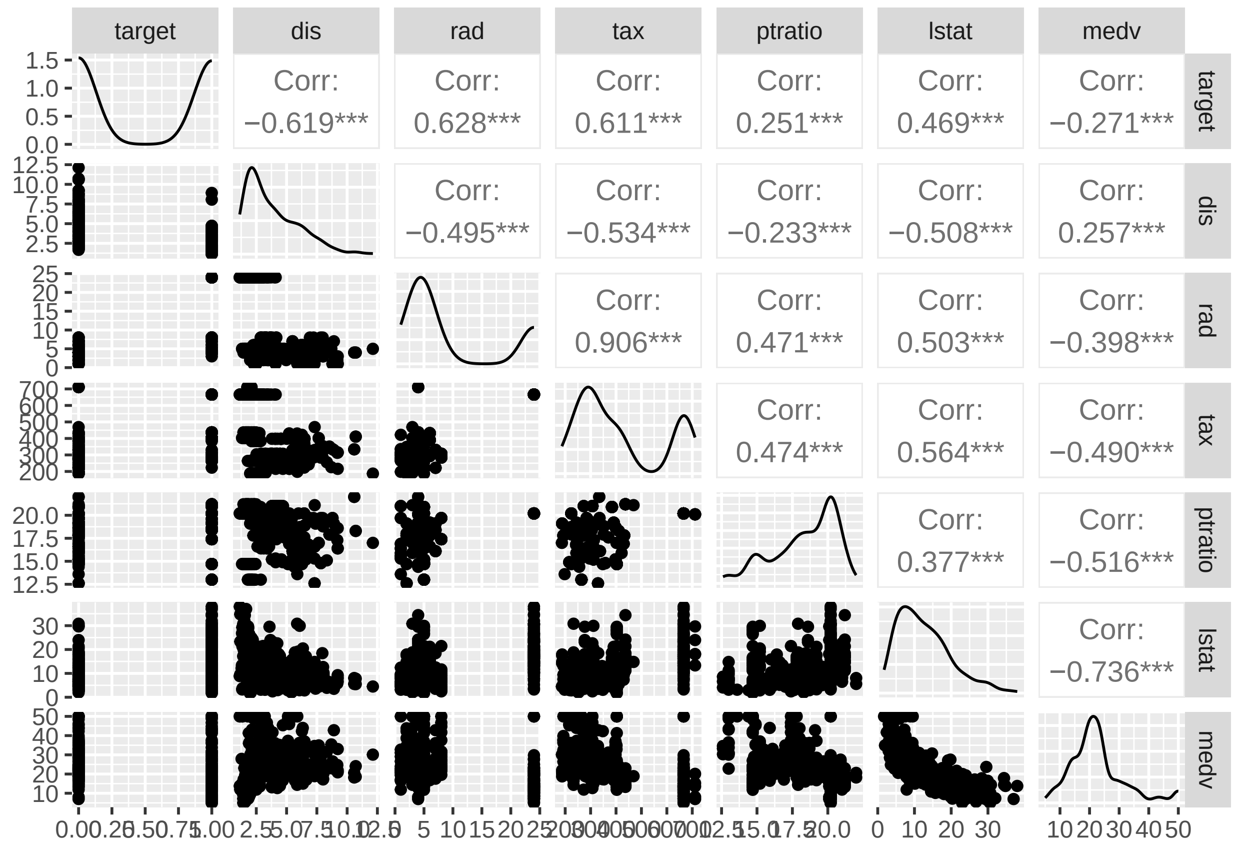


Interestingly, it seems that every field is significantly correlated except for `chas`. The ones with negative correlation are `zn` and `rm` while `indus`, `nox`, and `age` are positively correlated.

Implication wise, this seems to say that less residential space/rooms means more crime while more non-retail acres per suburb, older occupied buildings, and higher nitrogen oxide means more crime as well.

These fields are also pretty correlated with one another for the most part which may serve as an issue for our model.

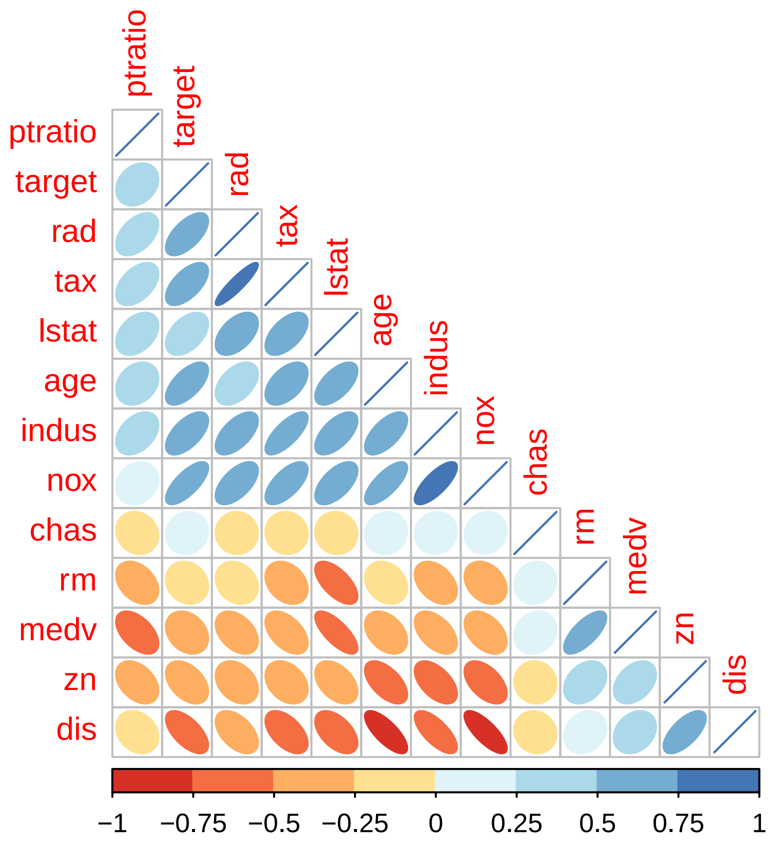
What do the relationships look like for the rest of the fields?



All of these fields are correlated with our `target` – `dis` and `medv` are the only ones with negative impact while the rest are positive. Additionally, they're all once again correlated with one another.

Implication wise, this means that the closer the area to employment centers and lower value of homes means more crime while more access to highways, higher tax rates, higher teacher-student ratios, and more lower status population also indicate high crime.

To view a more concise correlation analysis overall:



Looking at this, we can see an extremely strong correlation between `rad` and `tax` as well as a slightly less but still strong correlation between `indus` and `nox`.

Keeping this information in mind as we move closer to creating our model, we'll move to the next step of preparing our data.

## Data Preparation

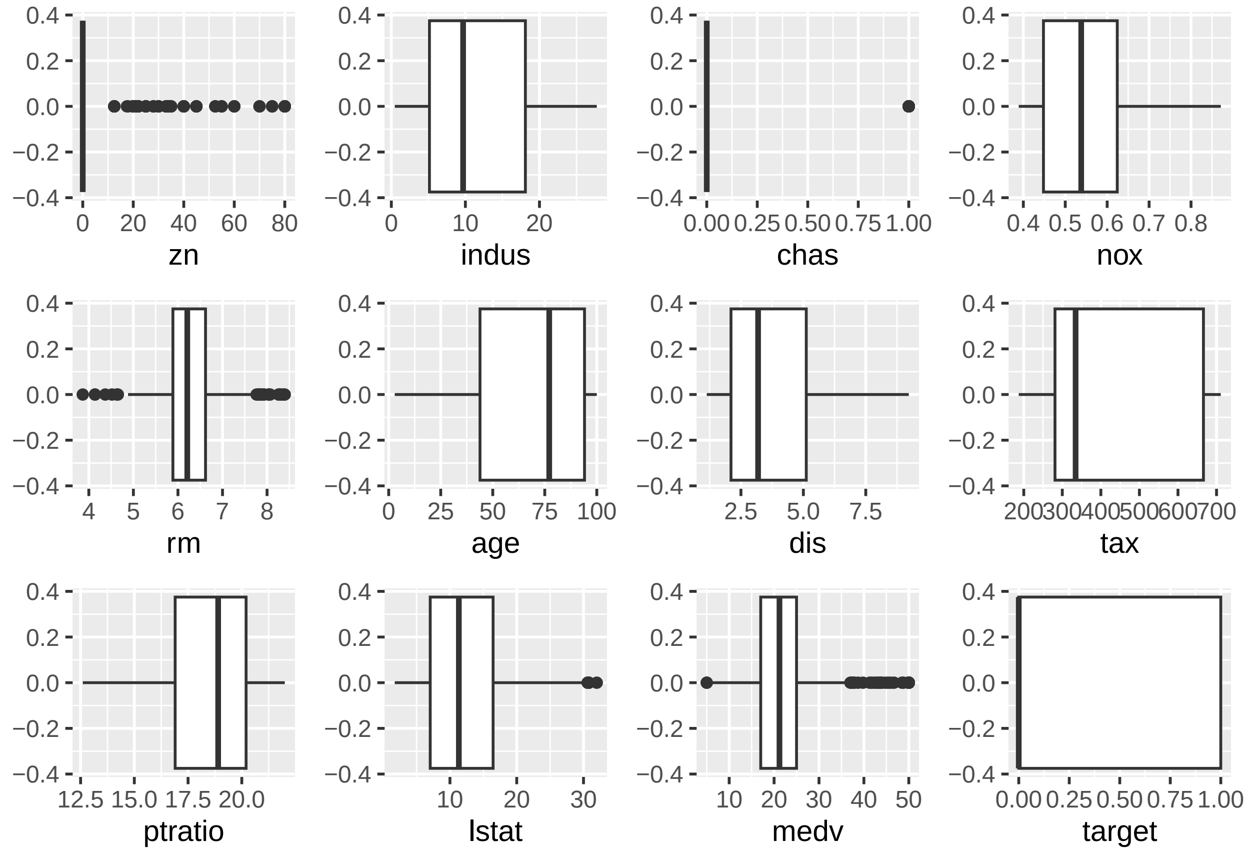
### Removals

To start, we will remove the `rad` field due to its strong correlation with `tax`. `rad` was chosen over `tax` because the latter visually looks a little more normal than the other and might be better suited for model use.

```
##   zn indus chas   nox    rm   age    dis tax ptratio lstat medv target
## 1  0 19.58    0 0.605 7.929  96.2 2.0459 403    14.7  3.70 50.0      1
## 2  0 19.58    1 0.871 5.403 100.0 1.3216 403    14.7 26.82 13.4      1
## 3  0 18.10    0 0.740 6.485 100.0 1.9784 666    20.2 18.85 15.4      1
## 4 30  4.93    0 0.428 6.393   7.8 7.0355 300    16.6  5.19 23.7      0
## 5  0  2.46    0 0.488 7.155  92.2 2.7006 193    17.8  4.82 37.9      0
## 6  0  8.56    0 0.520 6.781  71.3 2.8561 384    20.9  7.67 26.5      0
```

### Outliers

There are some pretty extreme outliers scattered throughout the `zn`, `rm`, `dis`, `lstat`, and `medv` fields (see the boxplot in the previous section). To account for these, we will use the median of the data again to replace these outliers if they are more than three standard deviations from the mean. This means replacing a total of 26 fields.



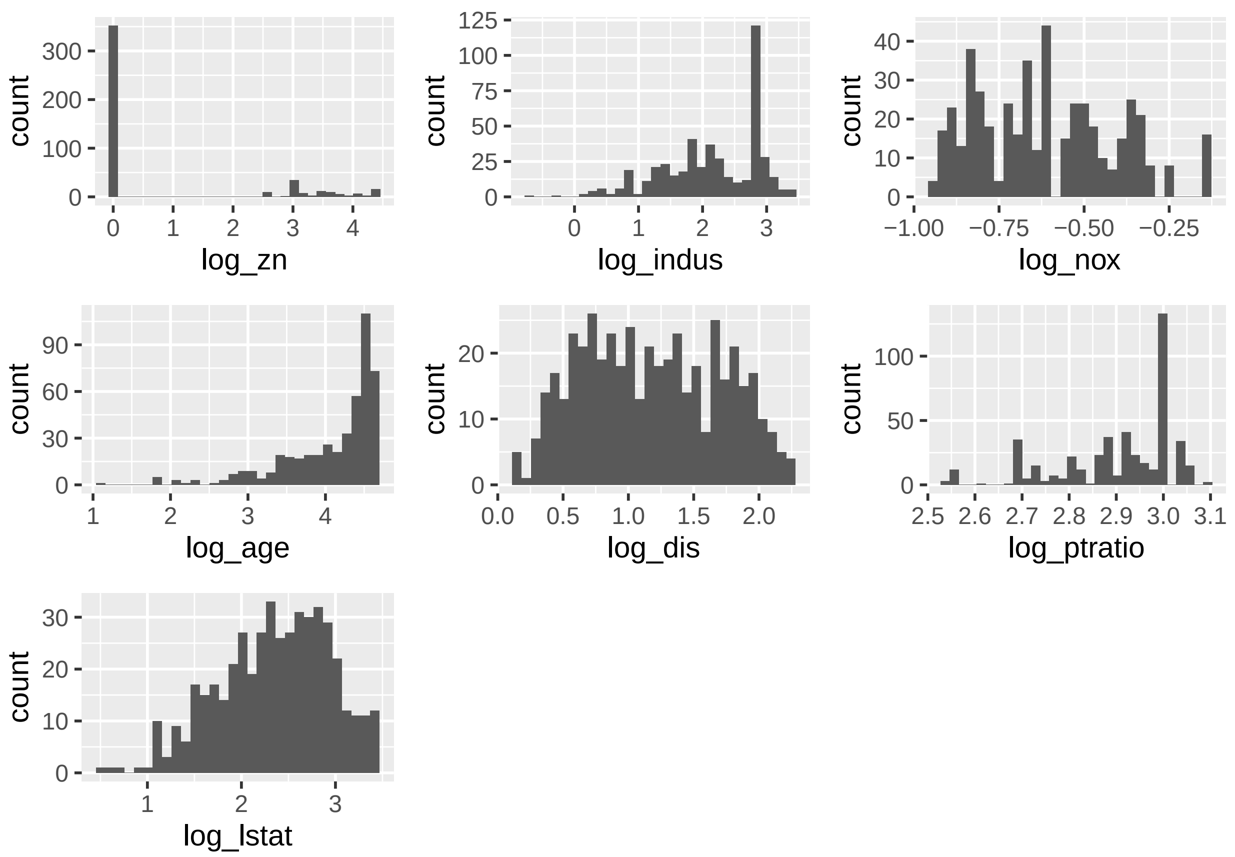
While there are still some outliers, the boxplots look a lot cleaner; it should be noted that for `medv`, although we tried to impute outliers, it seems the max value 50 was just below the threshold so nothing in this field was actually changed.

## Transform Non-Normal Variables

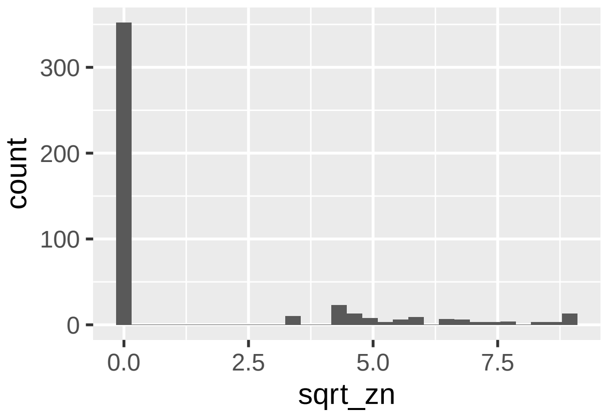
The last alteration before modeling is ensuring that our variables are normal by transforming the ones that don't seem to have much of normal distribution. The fields with distributions that aren't normal are:

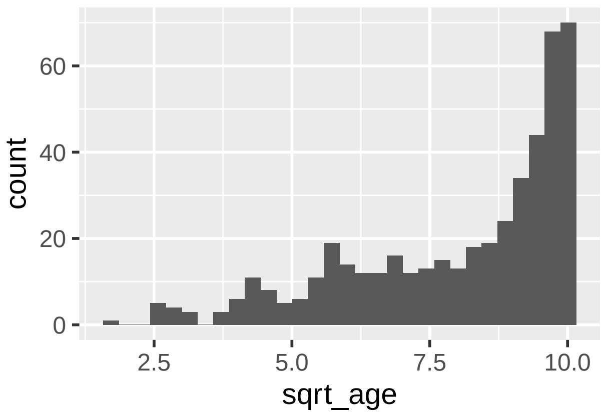
- `zn`
- `indus`
- `nox`
- `age`
- `dis`
- `ptratio`
- `lstat`

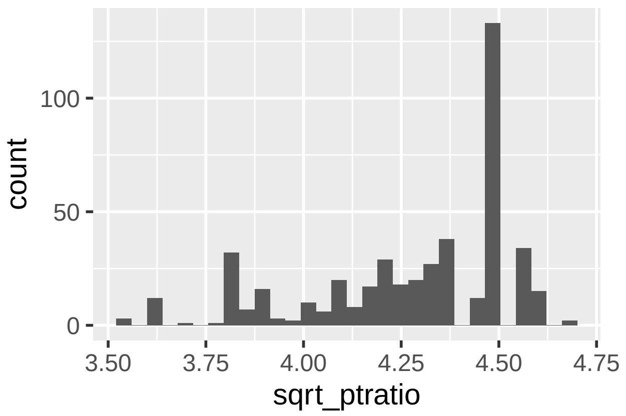
We'll try transforming these with `log` first and if that doesn't work, then we'll `sqrt` it.



It looks like this fixed a few variables, however `zn`, `age`, and `ptratio` still aren't vern normalized. Let's trying using `sqrt` on them.







This also wasn't super helpful. Given not every piece of data can be normalized, we'll opt to keep these versions of the three above variables:

- `zn` -> `log_zn`
- `age` -> `log_age`
- `ptratio` -> `sqrt_ptratio`

## Build Models

Before doing anything, we will split the data into training and test sets with a 70/30 split.

We'll go through two sets of models:

- Model 1: Start from using all the coefficients as is and only use the transformed ones if they don't seem to have a solid impact on the model
- Model 2: Start with all normalized (to the best of our ability) variables and select from there

### Model 1A

This first model will use all the fields pre-transformed ones.

```
##
## Call:
## glm(formula = target ~ zn + indus + chas + nox + rm + age + dis +
##     tax + ptratio + lstat + medv, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3382  -0.3970  -0.0316   0.1255   3.1868
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -39.219298   7.029128  -5.580 2.41e-08 ***
## zn           -0.047379   0.034617  -1.369 0.171106
## indus        -0.123558   0.056320  -2.194 0.028244 *
## chas          1.294272   0.802105   1.614 0.106616
## nox          48.570777   8.583641   5.659 1.53e-08 ***
## rm           -0.472184   0.659200  -0.716 0.473807
## age           0.012046   0.013250   0.909 0.363308
## dis           0.740283   0.222931   3.321 0.000898 ***
## tax           0.006701   0.002139   3.133 0.001732 **
## ptratio       0.340162   0.119826   2.839 0.004528 **
## lstat         0.031071   0.060130   0.517 0.605343
## medv          0.197751   0.062067   3.186 0.001442 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 451.97  on 326  degrees of freedom
## Residual deviance: 167.74  on 315  degrees of freedom
## AIC: 191.74
##
## Number of Fisher Scoring iterations: 7
```

**Coefficient Evaluation**

Looking at the model's coefficients and whether they had a positive or negative impact, `zn`, `indus`, `rm`, and the intercept being negative imply that more residential space and more non-retail businesses would contribute less to crime while just everything at 0 would mean that a neighborhood would likely not have high crime. These all have a lower amount of impact than some of the other variables though as an observation.

`nox` has a huge impact with `chas` also being pretty positive – this would imply that high nitrogen oxide as well as bordering the Charles River is a good indicator for high crime.

**Significance Evaluation & Performance**

`nox` and the intercept hold the strongest level of significance at 0, `dis`, `tax`, `ptratio`, and `medv` have slightly less significance at 0.001, and `indus` is the least level at 0.01. This leaves `zn`, `chas`, `rm`, `age`, and `lstat` – we will opt to remove these five in the next iteration.

With an AIC of 191.74 and residual deviance of 167.74, we'll use this as a baseline to compare to as we iterate on the model.

### Model 1B

```
##
## Call:
## glm(formula = target ~ indus + nox + dis + tax + ptratio + medv,
##     family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.32203  -0.40502  -0.09586   0.11808   2.82516
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -37.113683   6.033528  -6.151 7.69e-10 ***
## indus        -0.106931   0.052604  -2.033 0.042078 *
## nox          47.970396   8.185573   5.860 4.62e-09 ***
## dis           0.481869   0.182737   2.637 0.008365 **
## tax           0.005476   0.001891   2.896 0.003777 **
## ptratio       0.294560   0.108972   2.703 0.006870 **
## medv          0.132383   0.034845   3.799 0.000145 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 451.97  on 326  degrees of freedom
## Residual deviance: 175.50  on 320  degrees of freedom
## AIC: 189.5
##
## Number of Fisher Scoring iterations: 7
```

**Coefficient Evaluation**
Similar to Model 1A, `indus` is the only negative field meaning the higher it is, the less likely the area would contribute to crime. On the other side, `nox` still holds the most strength in impacting crime in a positive way, meaning the higher these values, the more likely there is to be a high level of crime.

**Significance Evaluation & Performance**
`indus` is the least significant, however it's still at 0.01 so still pretty strong; the rest of the variables are more significant and that's a promising sign. The AIC has improved at 189.5 vs. 191.74 while the residual deviance went up at 175.50 vs. 167.74.

### Model 2A

This model will use all the fields, defaulting to the ones that are normalized/transformed.

```
##
## Call:
## glm(formula = target ~ log_zn + log_indus + chas + log_nox +
##     rm + log_age + log_dis + tax + sqrt_ptratio + log_lstat +
##     medv, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9598  -0.3357  -0.0455   0.1954   3.1782
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -11.885282   7.515405  -1.581 0.113774
## log_zn        -0.338818   0.248764  -1.362 0.173195
## log_indus      0.016369   0.514918   0.032 0.974640
## chas           1.010604   0.794726   1.272 0.203502
## log_nox       26.159817   4.371495   5.984 2.17e-09 ***
## rm            -0.190651   0.592720  -0.322 0.747714
## log_age        0.284534   0.626155   0.454 0.649531
## log_dis        4.212167   1.046162   4.026 5.67e-05 ***
## tax            0.008038   0.002238   3.591 0.000330 ***
## sqrt_ptratio   2.985919   1.077792   2.770 0.005599 **
## log_lstat      0.772069   0.763270   1.012 0.311764
## medv           0.239124   0.067443   3.546 0.000392 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 451.97  on 326  degrees of freedom
## Residual deviance: 165.42  on 315  degrees of freedom
## AIC: 189.42
##
## Number of Fisher Scoring iterations: 7
```

**Coefficient Evaluation**

Looking at the model's coefficients and whether they had a positive or negative impact, `zn`, `rm`, and the intercept are all negative, however unlike Model 1's iteration, `indus` (`log_indus` in this case) is positive interestingly enough. Its weight though is much less than Model 1's so perhaps it's more of a borderline variable than we thought previously.

In terms of positive strength, `log_nox`, `log_dis`, and `sqrt_ptratio` are all quite high and contribute greatly to the likelihood that an area has high crime similar to Model 1, the additions here being that a longer distance from Boston's employment centers and a higher pupil-teacher ratio contribute to higher crime.

**Significance Evaluation & Performance**

Only five variables made it past a level of significance: `log_nox`, `log_dis`, `tax`, and `medv` have the highest levels of significance at 0. while `sqrt_ptratio` follows at 0.001. Given the rest are a bit far from being significant, we'll opt to remove them from the next iteration.

With an AIC of 189.42 and residual deviance of 165.42, we'll use this as a baseline to compare to as we iterate on the model.

**Model 2B**

```
##
## Call:
## glm(formula = target ~ log_nox + log_dis + tax + sqrt_ptratio +
##     medv, family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.82717  -0.34807  -0.07707   0.20294   2.86846
##
## Coefficients:
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -6.292691   4.455029  -1.412 0.157805
## log_nox      25.618818   3.949922   6.486 8.82e-11 ***
## log_dis       3.227369   0.857945   3.762 0.000169 ***
## tax           0.006542   0.001966   3.327 0.000877 ***
## sqrt_ptratio  2.798448   0.957136   2.924 0.003458 **
## medv          0.167597   0.038842   4.315 1.60e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 451.97  on 326  degrees of freedom
## Residual deviance: 171.33  on 321  degrees of freedom
## AIC: 183.33
##
## Number of Fisher Scoring iterations: 7
```

#### Coefficient Evaluation

Looking at the model's coefficients and whether they had a positive or negative impact, the intercept is now the only negative impactor.

In terms of positive strength, `log_nox`, `log_dis`, and `sqrt_ptratio` are still quite high and contribute greatly to the likelihood that an area has high crime with `tax` having a very low value.

#### Significance Evaluation & Performance

All variables are statistically significant now minus the intercept, however there's little we can do about that. `sqrt_ptratio` is slightly less significant than the other variables, but at that strength, it's negligible.

Comparing performance, AIC improved going from 189.42 to 183.33, but residual deviance increased from 165.42 to 171.33, we'll use this as a baseline to compare to as we iterate on the model.

## Select Models

### Confusion Matrices

First, we'll take a look at confusion matrices for each of the models.

```r
# if the prediction is >= 0.5, then we would predict 1 for that row, otherwise 0
test$model1a <- ifelse(predict.glm(model1a, test, "response") >= 0.5, 1, 0)

# create the confusion matrix
cm1a <- confusionMatrix(factor(test$model1a), factor(test$target), "1")
results <- tibble(Model = "Model #1", Accuracy=cm1a$byClass[11], F1 = cm1a$byClass[7],
                  Deviance= model1a$deviance,
                  R2 = 1 - model1a$deviance / model1a$null.deviance,
                  AIC= model1a$aic)
cm1a
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 58 12
##          1  5 64
##
##                Accuracy : 0.8777
##                  95% CI : (0.8114, 0.9271)
##     No Information Rate : 0.5468
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7556
##
##  Mcnemar's Test P-Value : 0.1456
##
##             Sensitivity : 0.8421
##             Specificity : 0.9206
##          Pos Pred Value : 0.9275
##          Neg Pred Value : 0.8286
##              Prevalence : 0.5468
##          Detection Rate : 0.4604
##    Detection Prevalence : 0.4964
##       Balanced Accuracy : 0.8814
##
##        'Positive' Class : 1
##
```

```r
# if the prediction is >= 0.5, then we would predict 1 for that row, otherwise 0
test$model1b <- ifelse(predict.glm(model1b, test, "response") >= 0.5, 1, 0)

# create the confusion matrix
cm1b <- confusionMatrix(factor(test$model1b), factor(test$target), "1")
results <- tibble(Model = "Model #1", Accuracy=cm1b$byClass[11], F1 = cm1b$byClass[7],
                  Deviance= model1b$deviance,
                  R2 = 1 - model1b$deviance / model1b$null.deviance,
                  AIC= model1b$aic)
cm1b
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 56 16
##          1  7 60
##
##                Accuracy : 0.8345
##                  95% CI : (0.7621, 0.8921)
##     No Information Rate : 0.5468
##     P-Value [Acc > NIR] : 6.848e-13
##
##                   Kappa : 0.6702
##
##  Mcnemar's Test P-Value : 0.09529
##
##             Sensitivity : 0.7895
##             Specificity : 0.8889
##          Pos Pred Value : 0.8955
##          Neg Pred Value : 0.7778
##              Prevalence : 0.5468
##          Detection Rate : 0.4317
##    Detection Prevalence : 0.4820
##       Balanced Accuracy : 0.8392
##
##        'Positive' Class : 1
##
```

```
# if the prediction is >= 0.5, then we would predict 1 for that row, otherwise 0
test$model2a <- ifelse(predict.glm(model2a, test, "response") >= 0.5, 1, 0)

# create the confusion matrix
cm2a <- confusionMatrix(factor(test$model2a), factor(test$target), "1")
results <- tibble(Model = "Model #1", Accuracy=cm2a$byClass[11], F1 = cm2a$byClass[7],
                  Deviance= model2a$deviance,
                  R2 = 1 - model2a$deviance / model2a$null.deviance,
                  AIC= model2a$aic)
cm2a
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 56 15
##          1  7 61
##
##                Accuracy : 0.8417
##                  95% CI : (0.7702, 0.8981)
##     No Information Rate : 0.5468
##     P-Value [Acc > NIR] : 1.599e-13
##
##                   Kappa : 0.6841
##
##  Mcnemar's Test P-Value : 0.1356
##
##             Sensitivity : 0.8026
##             Specificity : 0.8889
##          Pos Pred Value : 0.8971
##          Neg Pred Value : 0.7887
##              Prevalence : 0.5468
##          Detection Rate : 0.4388
##    Detection Prevalence : 0.4892
##       Balanced Accuracy : 0.8458
##
##        'Positive' Class : 1
##
```

```
# if the prediction is >= 0.5, then we would predict 1 for that row, otherwise 0
test$model2b <- ifelse(predict.glm(model2b, test, "response") >= 0.5, 1, 0)

# create the confusion matrix
cm2b <- confusionMatrix(factor(test$model2b), factor(test$target), "1")
```

```
results <- tibble(Model = "Model #1", Accuracy=cm2b$byClass[11], F1 = cm2b$byClass[7],
                  Deviance= model2b$deviance,
                  R2 = 1 - model2b$deviance / model2b$null.deviance,
                  AIC= model2b$aic)
cm2b
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0  1
##            0 55 15
##            1  8 61
##
##                Accuracy : 0.8345
##                  95% CI : (0.7621, 0.8921)
##     No Information Rate : 0.5468
##     P-Value [Acc > NIR] : 6.848e-13
##
##                   Kappa : 0.6693
##
##  Mcnemar's Test P-Value : 0.2109
##
##             Sensitivity : 0.8026
##             Specificity : 0.8730
##          Pos Pred Value : 0.8841
##          Neg Pred Value : 0.7857
##              Prevalence : 0.5468
##          Detection Rate : 0.4388
##    Detection Prevalence : 0.4964
##       Balanced Accuracy : 0.8378
##
##        'Positive' Class : 1
##
```

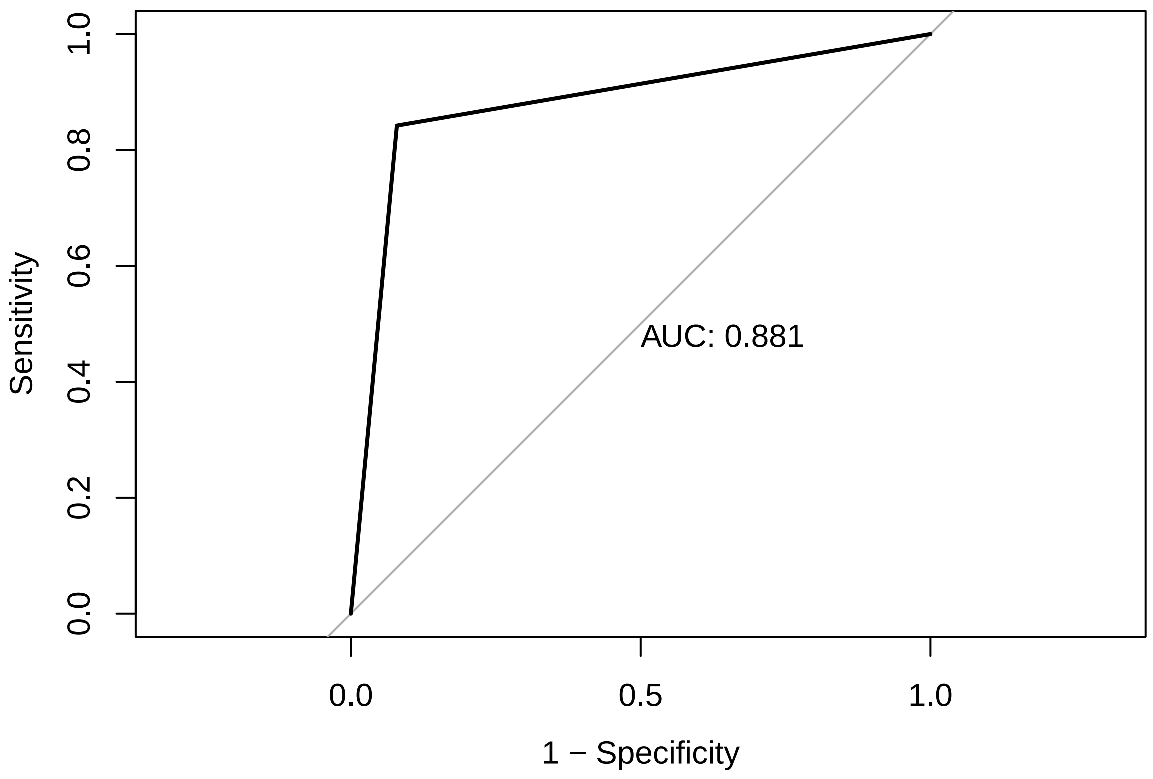
### ROC

Now with all of these matrices, we'll look at ROC curves.

```
print('Model 1A ROC Curve')
```

```
## [1] "Model 1A ROC Curve"
```

```
roc(test[["target"]], test[["model1a"]], plot = TRUE, legacy.axes = TRUE, print.auc = TRUE)
```

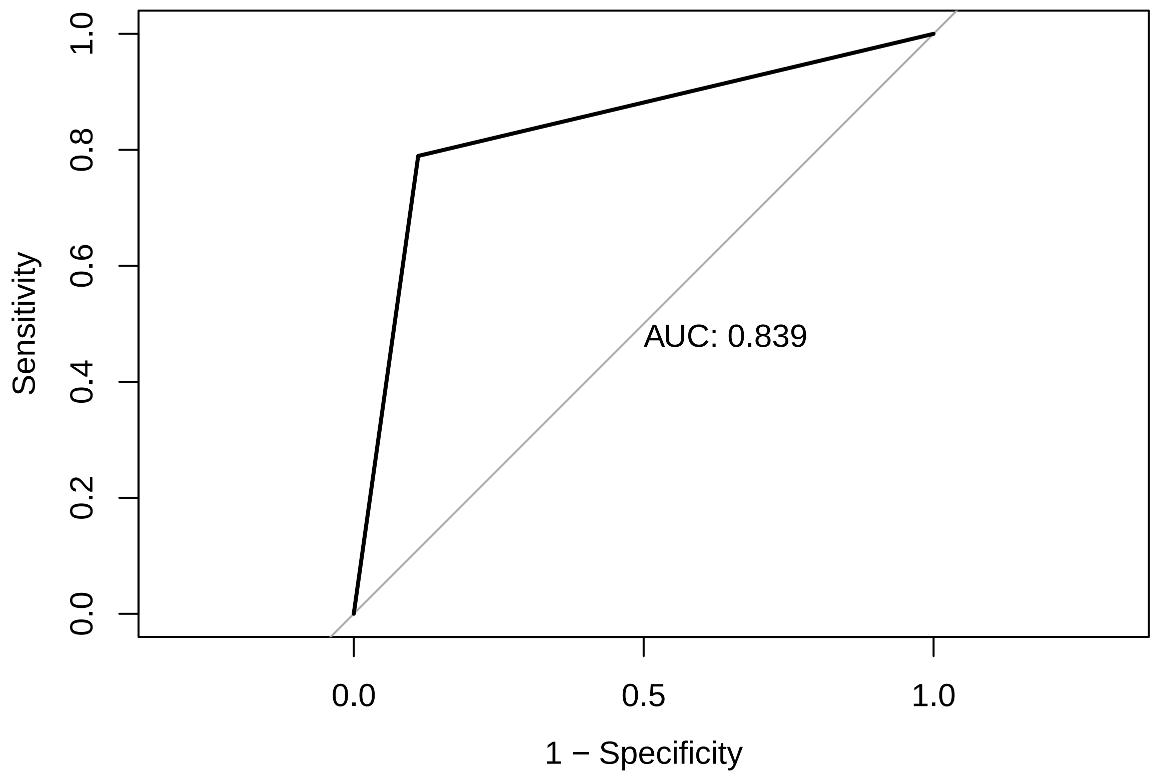


```
##
## Call:
## roc.default(response = test[["target"]], predictor = test[["model1a"]],     plot = TRUE, legacy.axes
##
## Data: test[["model1a"]] in 63 controls (test[["target"]] 0) < 76 cases (test[["target"]] 1).
## Area under the curve: 0.8814
```

```
print('Model 1B ROC Curve')
```

```
## [1] "Model 1B ROC Curve"
```

```
roc(test[["target"]], test[["model1b"]], plot = TRUE, legacy.axes = TRUE, print.auc = TRUE)
```

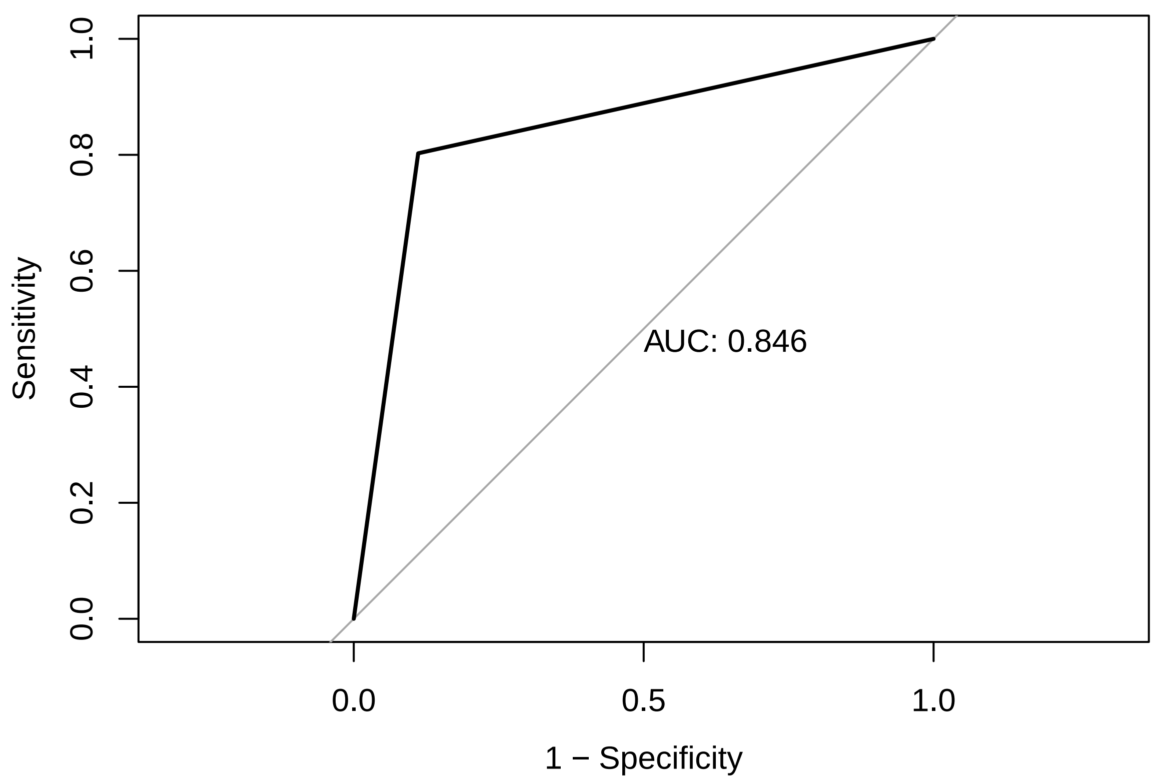


```
##
## Call:
## roc.default(response = test[["target"]], predictor = test[["model1b"]],     plot = TRUE, legacy.axes
##
## Data: test[["model1b"]] in 63 controls (test[["target"]] 0) < 76 cases (test[["target"]] 1).
## Area under the curve: 0.8392
```

```
print('Model 2A ROC Curve')
```

```
## [1] "Model 2A ROC Curve"
```

```
roc(test[["target"]], test[["model2a"]], plot = TRUE, legacy.axes = TRUE, print.auc = TRUE)
```

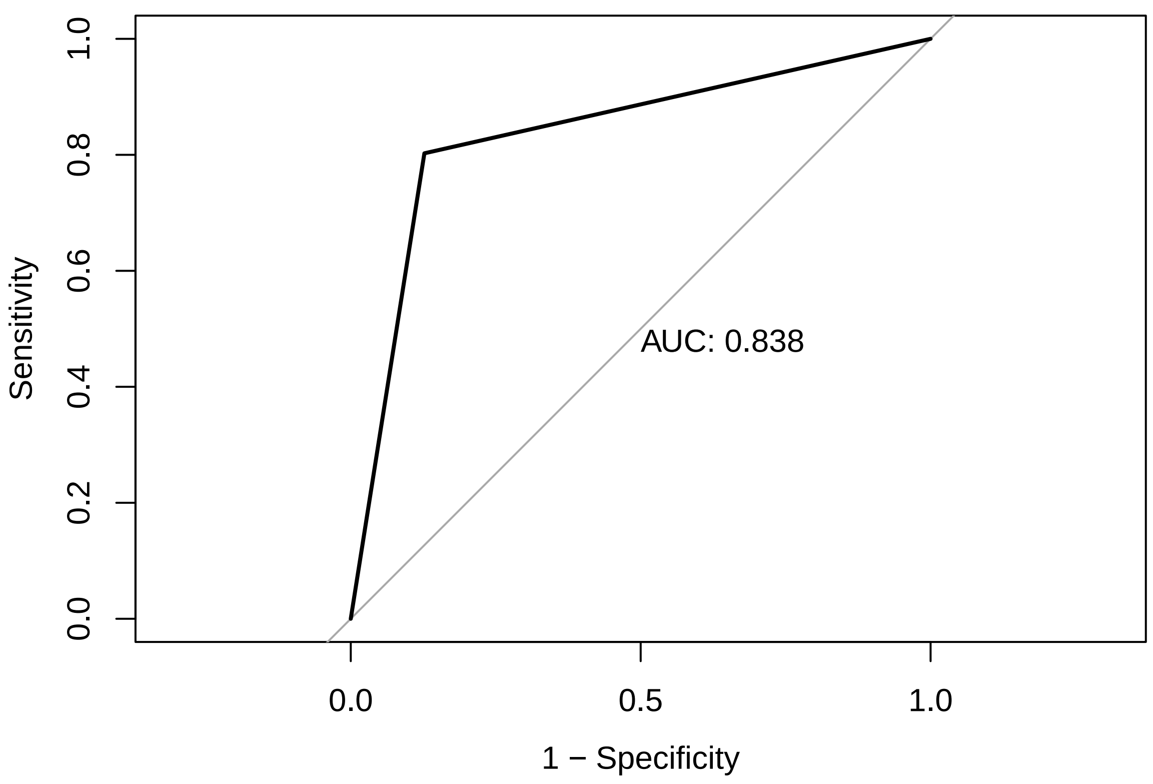


```
##
## Call:
## roc.default(response = test[["target"]], predictor = test[["model2a"]],     plot = TRUE, legacy.axes
##
## Data: test[["model2a"]] in 63 controls (test[["target"]] 0) < 76 cases (test[["target"]] 1).
## Area under the curve: 0.8458
```

```
print('Model 2B ROC Curve')
```

```
## [1] "Model 2B ROC Curve"
```

```
roc(test[["target"]], test[["model2b"]], plot = TRUE, legacy.axes = TRUE, print.auc = TRUE)
```



```
##
## Call:
## roc.default(response = test[["target"]], predictor = test[["model2b"]],     plot = TRUE, legacy.axes
##
## Data: test[["model2b"]] in 63 controls (test[["target"]] 0) < 76 cases (test[["target"]] 1).
## Area under the curve: 0.8378
```

**Overall Comparisons**

```
##          Residual.Deviance      AIC  Accuracy        F1        R2
## Model 1A          167.7381 191.7381 0.8813701 0.8827586 0.6288723
## Model 1B          175.4969 189.4969 0.8391813 0.8391608 0.6117056
## Model 2A          165.4241 189.4241 0.8457602 0.8472222 0.6339921
## Model 2B          171.3295 183.3295 0.8378237 0.8413793 0.6209262
```

Based on the above output:

- Residual Deviance: Model 2A had the lowest Residual Deviance with Model 1A close behind
- AIC: Model 2B had the best AIC
- Accuracy: Model 1A had the best Accuracy
- F1: Model 1A had the best F1 statistic
- Rˆ2: Model 2A had the highest Rˆ2 with Model 1A close behind

## Conclusion

Overall, it seems like Model 2A performed the best with good Residual Deviance and Rˆ2, decent Accuracy, AIC, and F1 statistic. While the model isn't perfect or better by a huge margin when compared against the other models, it seems like the most well-rounded one. This makes sense to us as it utilizes our modified/more normalized variables and doesn't remove a majority of variables like Model 2B.