Data 621 - Homework 3

Group 4 Layla Quinones, Ian Costello, Dmitriy Burtsev & Esteban Aramayo

11/7/2021

Overview

General Objective

For this assignment, we will be exploring, analyzing, and modeling data related to crime statistics for various areas of a major U.S. city. The primary objective is to understand how, or if, variable indicate whether crime in a particular area will be above or below the median crime rate for the entire city. The models will be binary logistic regression using combinations or constructions of the variables provided.

About the Data

- 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)

Libraries Used

We use pretty standard packages for this assignment, including the ever-useful `tidyverse`, `ggplot2`, and `caret`. New additions for this assignment include `VIM`, `DataExplorer`, and `broom`. **(Code Appendix 1.1)**

Data Exploration

As usual, our data are stored on GitHub at our team's main repository for easy access across team members **(Code Appendix 2.2)**. The variables are data type doubles, except for two variables: `chas` and `target`. While these are integer data types, they will be treated as categorical factors . Taking a peek into the data, we can get a sense of the distribution and structure of the data set **(Code Appendix 2.3)**. More visuals will be required to look deeper, however.

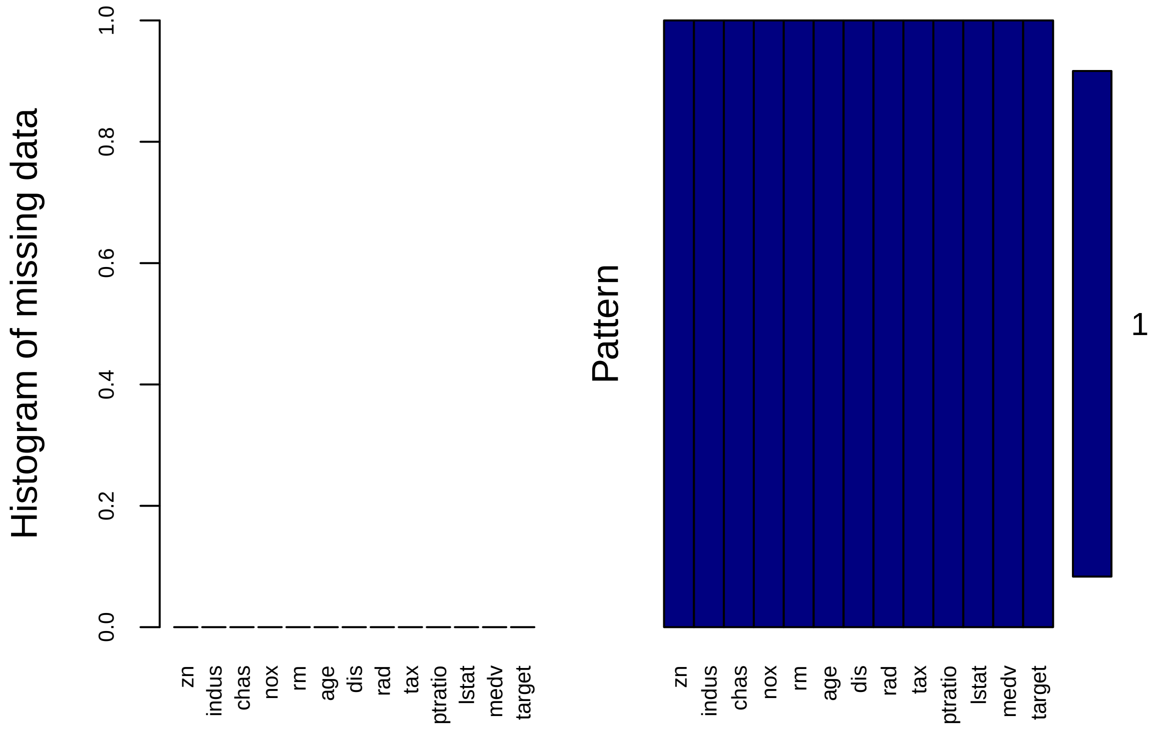
	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
zn	1	466	11.5772532	23.3646511	0.00000	5.3542781	0.0000000	0.0000	100.0000	100.0000	2.1768152	3.8135765	1.0823466
indus	2	466	11.1050215	6.8458549	9.69000	10.9082353	9.3403800	0.4600	27.7400	27.2800	0.2885450	-1.2432132	0.3171281
chas	3	466	0.0708155	0.2567920	0.00000	0.0000000	0.0000000	0.0000	1.0000	1.0000	3.3354899	9.1451313	0.0118957
nox	4	466	0.5543105	0.1166667	0.53800	0.5442684	0.1334340	0.3890	0.8710	0.4820	0.7463281	-0.0357736	0.0054045
rm	5	466	6.2906738	0.7048513	6.21000	6.2570615	0.5166861	3.8630	8.7800	4.9170	0.4793202	1.5424378	0.0326516
age	6	466	68.3675966	28.3213784	77.15000	70.9553476	30.0226500	2.9000	100.0000	97.1000	-0.5777075	-1.0098814	1.3119625
dis	7	466	3.7956929	2.1069496	3.19095	3.5443647	1.9144814	1.1296	12.1265	10.9969	0.9988926	0.4719679	0.0976026
rad	8	466	9.5300429	8.6859272	5.00000	8.6978610	1.4826000	1.0000	24.0000	23.0000	1.0102788	-0.8619110	0.4023678
tax	9	466	409.5021459	167.9000887	334.50000	401.5080214	104.5233000	187.0000	711.0000	524.0000	0.6593136	-1.1480456	7.7778214
ptratio	10	466	18.3984979	2.1968447	18.90000	18.5970588	1.9273800	12.6000	22.0000	9.4000	-0.7542681	-0.4003627	0.1017669
lstat	11	466	12.6314592	7.1018907	11.35000	11.8809626	7.0720020	1.7300	37.9700	36.2400	0.9055864	0.5033688	0.3289887
medv	12	466	22.5892704	9.2396814	21.20000	21.6304813	6.0045300	5.0000	50.0000	45.0000	1.0766920	1.3737825	0.4280200
target	13	466	0.4914163	0.5004636	0.00000	0.4893048	0.0000000	0.0000	1.0000	1.0000	0.0342293	-2.0031131	0.0231835

```
## Rows: 466
## Columns: 13
## $ zn      <dbl> 0, 0, 0, 30, 0, 0, 0, 0, 0, 80, 22, 0, 0, 22, 0, 0, 100, 20, 0~
## $ indus   <dbl> 19.58, 19.58, 18.10, 4.93, 2.46, 8.56, 18.10, 18.10, 5.19, 3.6~
## $ chas    <int> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ nox     <dbl> 0.605, 0.871, 0.740, 0.428, 0.488, 0.520, 0.693, 0.693, 0.515,~
## $ rm      <dbl> 7.929, 5.403, 6.485, 6.393, 7.155, 6.781, 5.453, 4.519, 6.316,~
## $ age     <dbl> 96.2, 100.0, 100.0, 7.8, 92.2, 71.3, 100.0, 100.0, 38.1, 19.1,~
## $ dis     <dbl> 2.0459, 1.3216, 1.9784, 7.0355, 2.7006, 2.8561, 1.4896, 1.6582~
## $ rad     <int> 5, 5, 24, 6, 3, 5, 24, 24, 5, 1, 7, 5, 24, 7, 3, 3, 5, 5, 24, ~
## $ tax     <int> 403, 403, 666, 300, 193, 384, 666, 666, 224, 315, 330, 398, 66~
## $ ptratio <dbl> 14.7, 14.7, 20.2, 16.6, 17.8, 20.9, 20.2, 20.2, 20.2, 16.4, 19~
## $ lstat   <dbl> 3.70, 26.82, 18.85, 5.19, 4.82, 7.67, 30.59, 36.98, 5.68, 9.25~
## $ medv    <dbl> 50.0, 13.4, 15.4, 23.7, 37.9, 26.5, 5.0, 7.0, 22.2, 20.9, 24.8~
## $ target  <int> 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0,~
```

Table 1. Glimpse of data structure (Code Appendix 2.4)

Missing Data Checks

Following standard procedure, we check for any missing data in the set **(Code Appendix 2.5)**. It appears there are no missing values as the following figures demonstrate.Again, we can see that most of the variables appear to be continuous. From the description of the predictors in the overview section of this document, we know that some of them can be treated as discrete and/or categorical. No columns with missing values were detected and all rows are complete.



```
##
##  Variables sorted by number of missings:
##  Variable Count
##        zn     0
##     indus     0
##      chas     0
##       nox     0
##        rm     0
##       age     0
##       dis     0
##       rad     0
##       tax     0
##   ptratio     0
##     lstat     0
##      medv     0
##    target     0
```

Figure 1. Plot missing values with VIM package

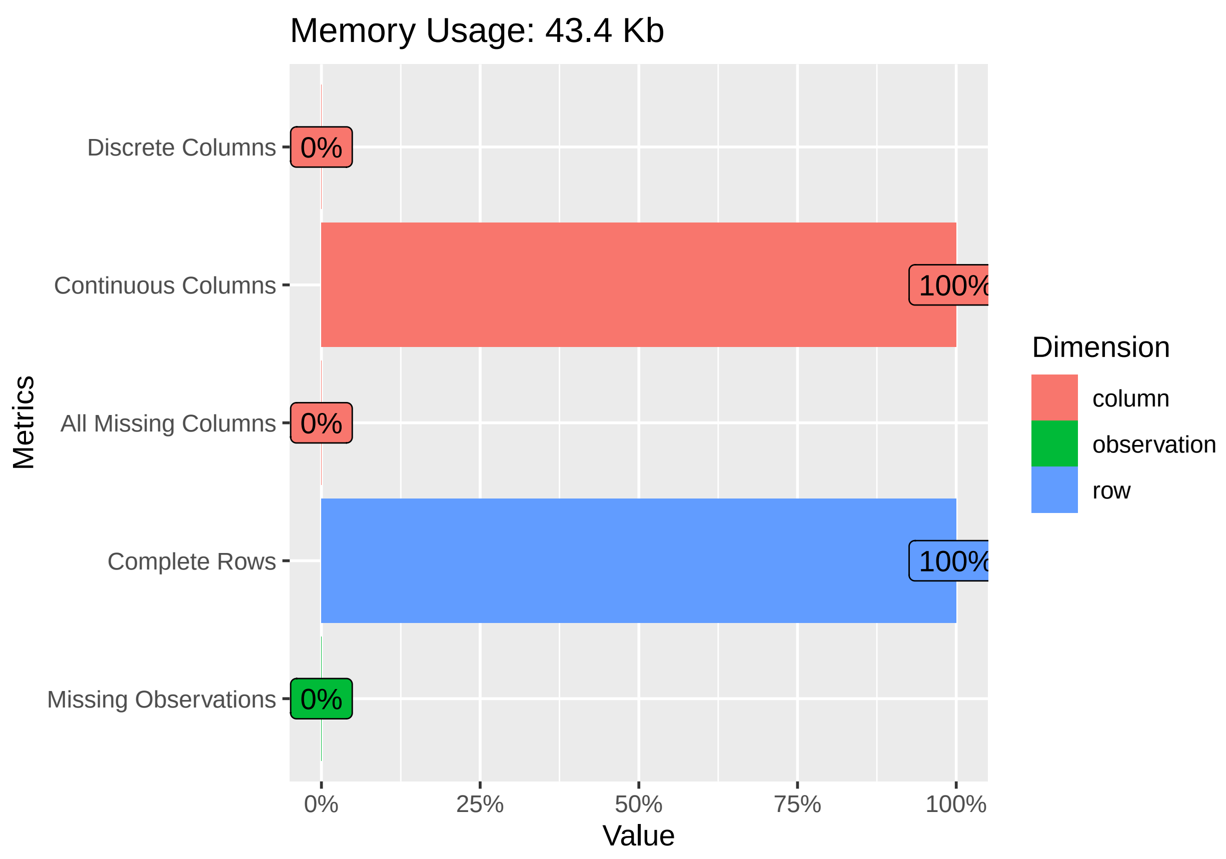


Figure 2. Plot missing values with DataExplorer package

Feature Histograms

For each of the variables, these histograms in figure 3 **(Code Appendix 2.6)** provide a nice overview of each feature, its variation, and paths for potential transformations later on for model construction. Histograms are a quick way to see the shape of the distributions for each feature. Of note are the normally distributed variables, median home value and number of rooms per dwelling. The remaining features appear quite skewed, especially land zoning, distance from employment, tax value, and distance to radial highways. We can also begin to see the affect of outliers that we'll have to account for.

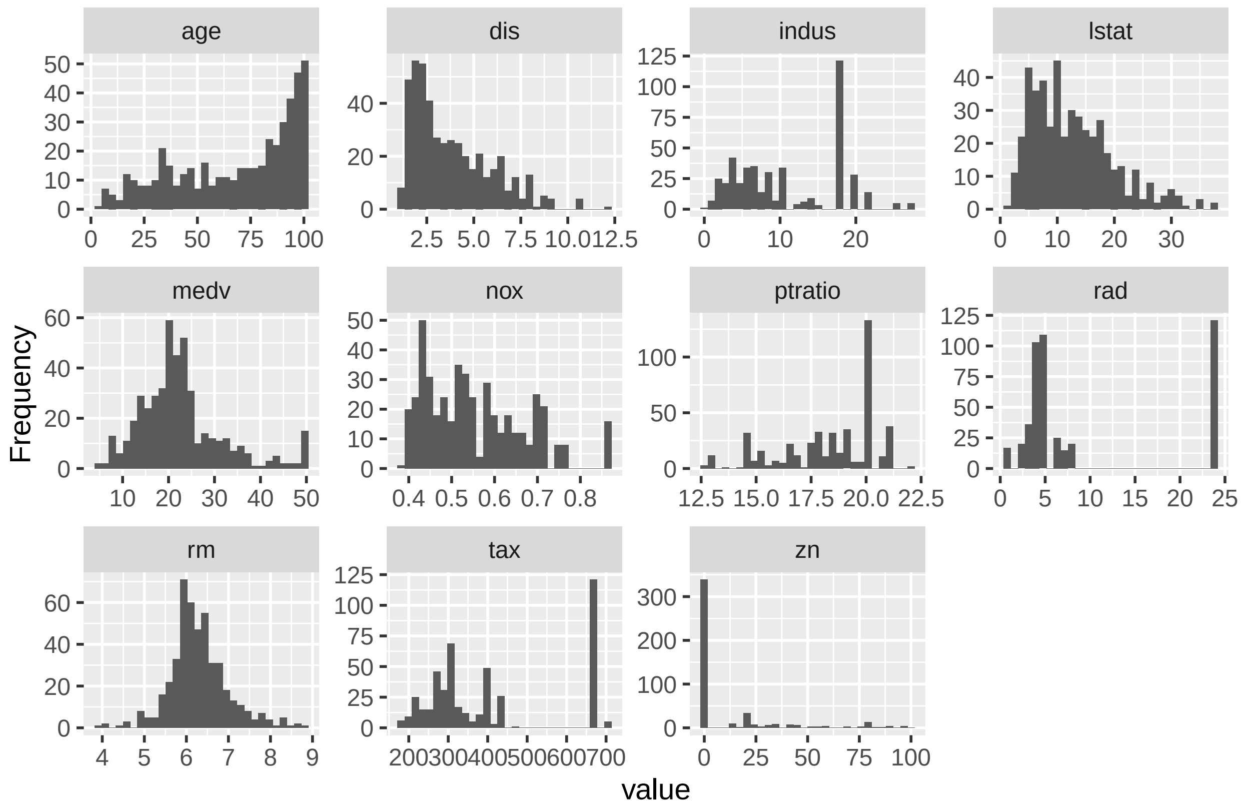


Figure 3. Feature histograms

Feature Boxplots

This box plot visualization (Figure 4) **(Code Appendix 2.7)** gives us an idea of the outliers we have in each variable, but does not give us a good sense of the distribution. We can use the histograms (Figure 3) above to interpret shape. We apply a log re-scaling to better compare the values across variables using a common scale and use notches to compare groups. If the notches of two boxes do not overlap, then this suggests that the medians are significantly different.

For the features we see significant outliers, we can decide to either throw out that variable out altogether and not consider it in our models or impute the outliers with median values. Before deciding on a course of action, we'll look at a few other things.

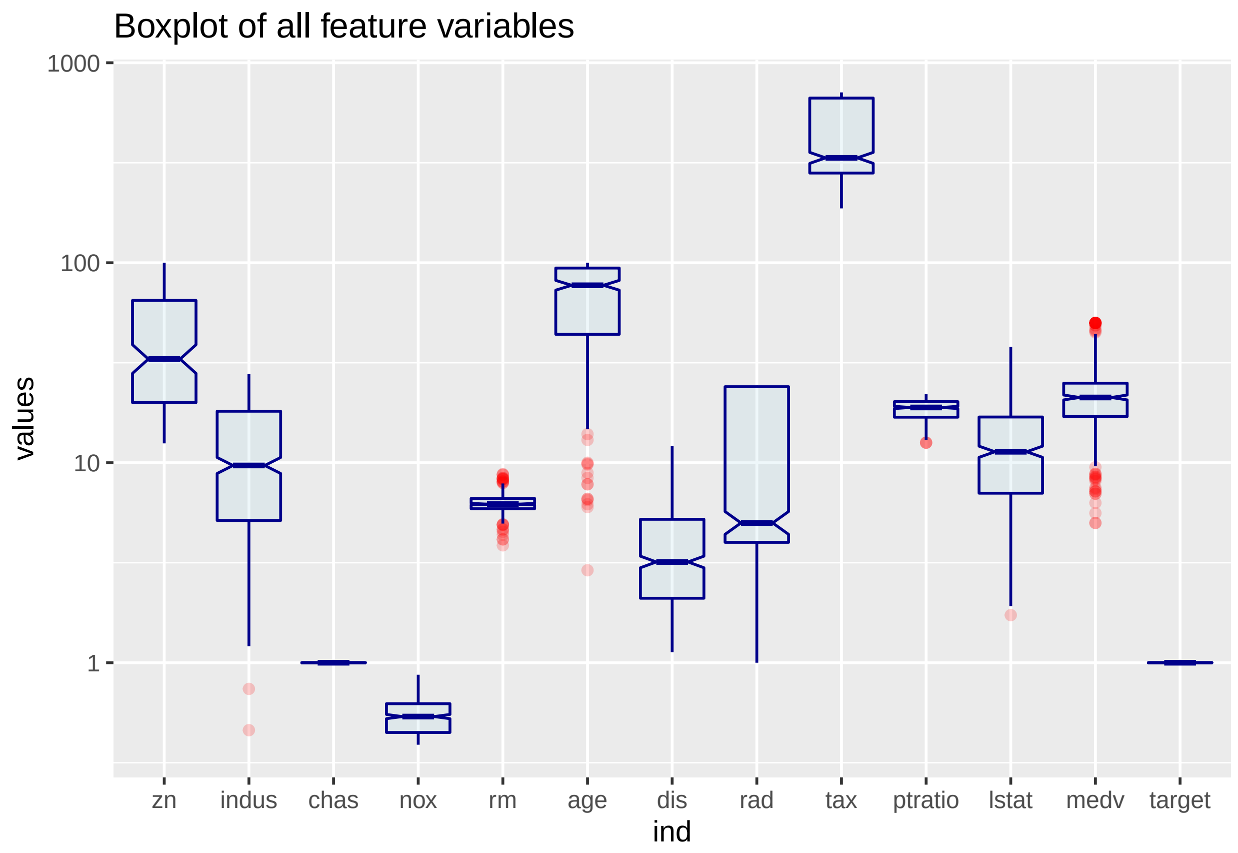


Figure 4. Feature box plots

Feature QQ Plots

The Quantile-Quantile, or QQ plots (figures 5 and 6) **(Code Appendix 2.8)** are used to visualize the deviation of the predictors compared to the normal distribution. It is rather normal to see tails on either side of the QQ as outliers will deviate significantly from the normal distribution. Too much and the feature will not be a good one to use as a predictor.

QQ Plots

Consistent with our other analysis of the features, `zn`, `rad`, and `tax` are not following the normal distribution enough to be helpful in our models, and with exception of the "chas" predictor, all other predictors will need to be transformed for linear regression. **(Code Appendix 2.8.1)**

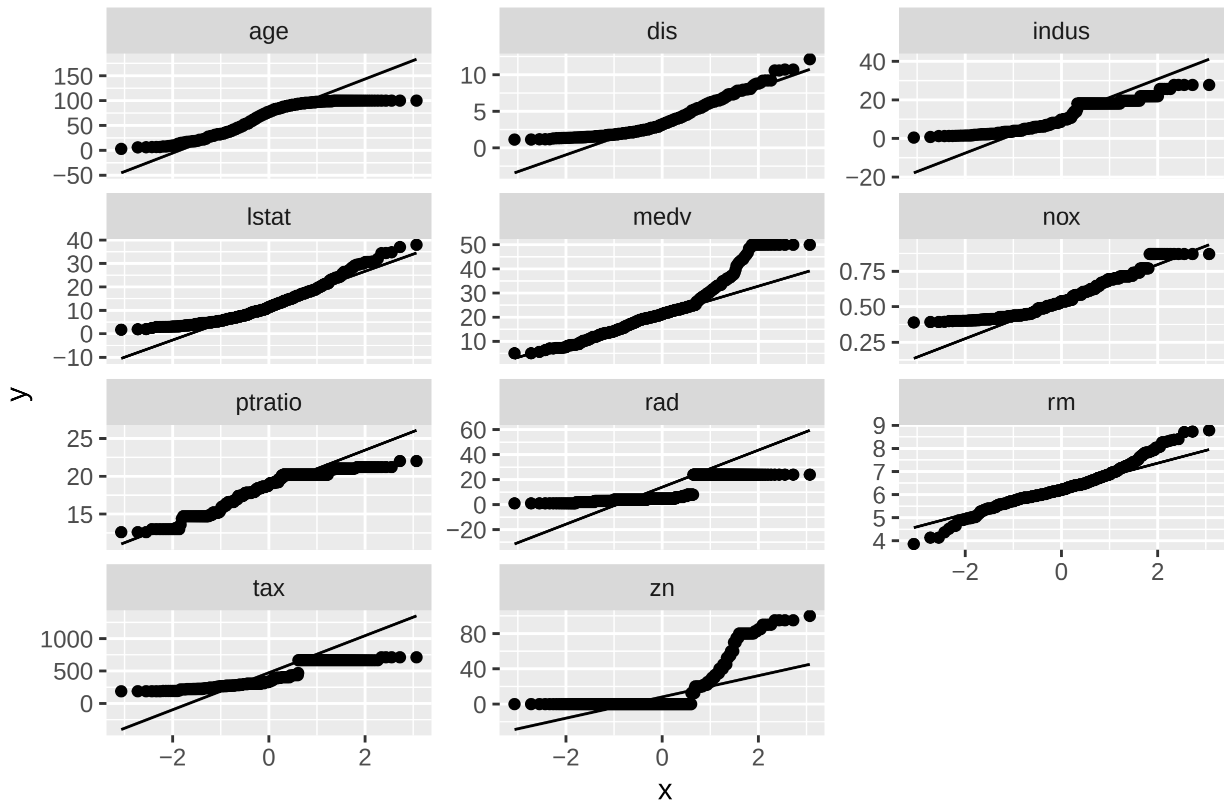


Figure 5. Feature QQ plot

Log QQ Plots

With the log transformation, the distributions look better now. So, as part of the data preparation we will transform the necessary predictors before we use them for the models. **(Code Appendix 2.8.2)**

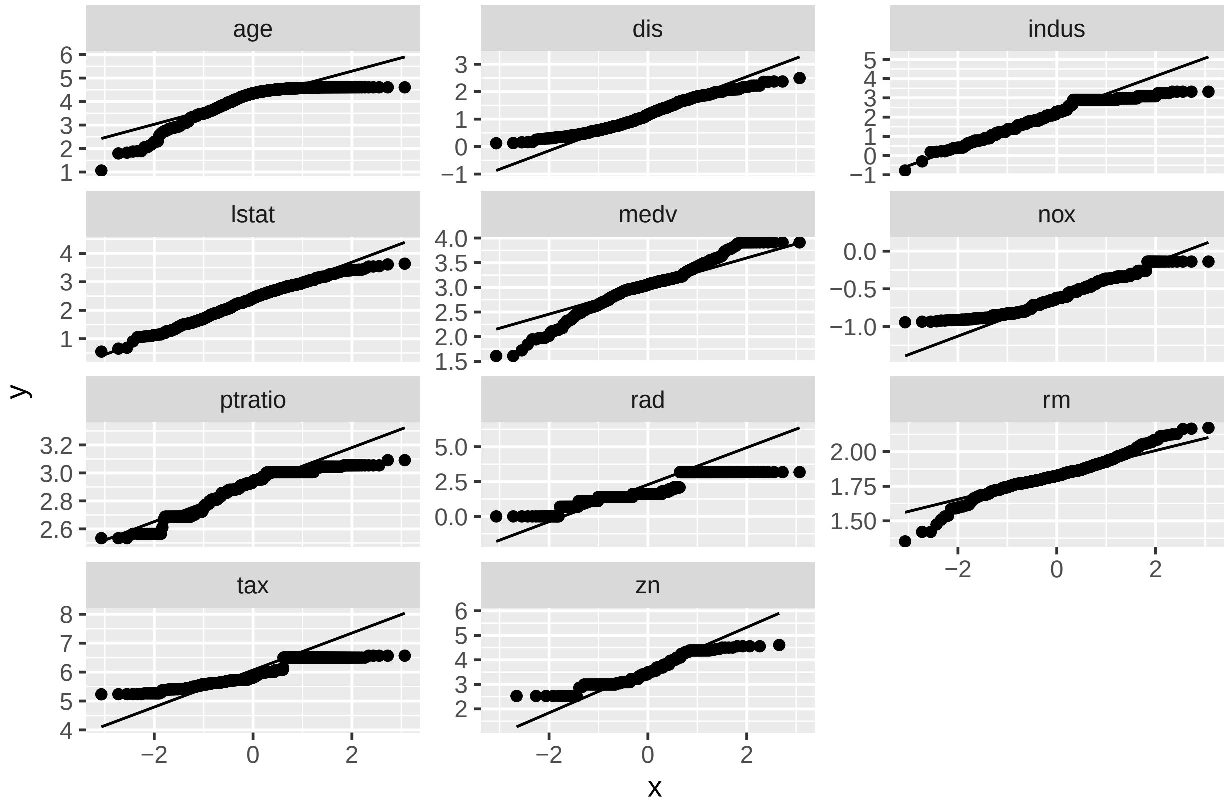


Figure 6. Feature QQ plots, log transformation

Correlation

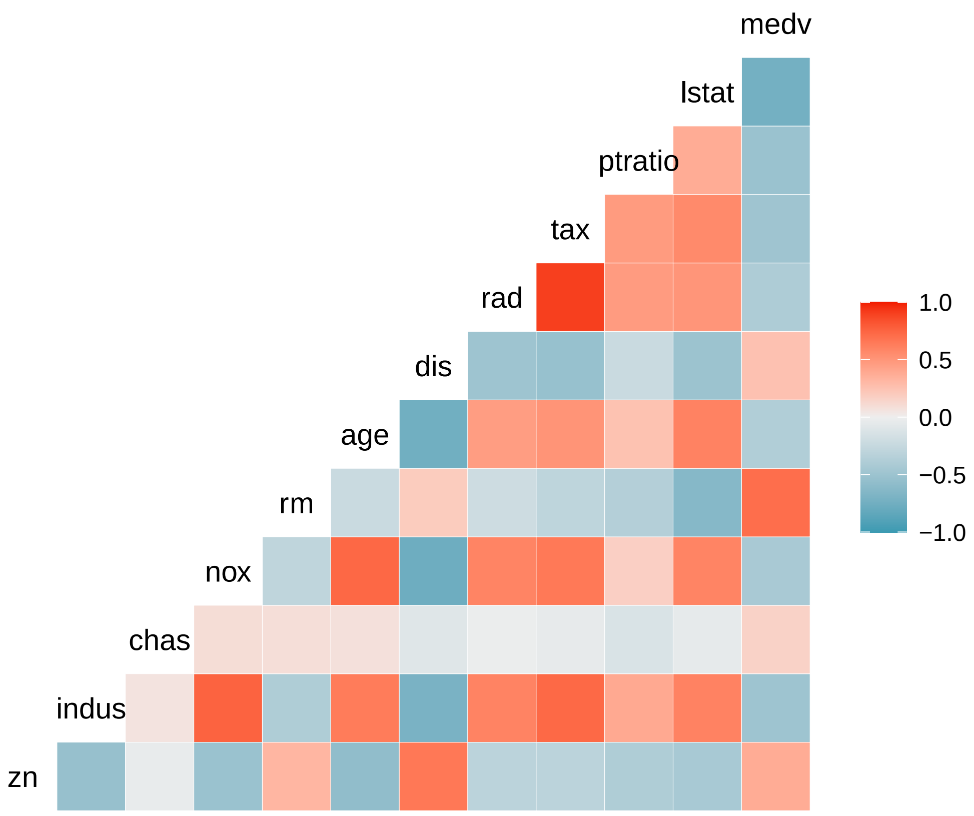


Figure 7. Correlation matrix

It is important to check for features which may also be correlated. Simply, having multiple features relate to themselves can cause overfitting, reduced *p* values, and strange variances in the data. To avoid this, we exclude one or more of the variables. In the correlation matrix (Figure 7) **(Code Appendix 2.9)**, we see that `tax` is very intertwined with two other variables besides the target, showing up as bright red. We'll take care when constructing our models not to use those.

Var1	Var2	Correlation
rad	tax	0.91
indus	nox	0.76
nox	age	0.74
indus	tax	0.73
nox	target	0.73*
rm	medv	0.71
age	target	0.63*
rad	target	0.03*
tax	target	0.61*

Table 3. Correlation results

Data Preparation

Outlier Value Analysis

Coming back to the outliers, let's analyze the variables with outliers and their relation to the `target` variable. **(Code Appendix 3.2.1)**

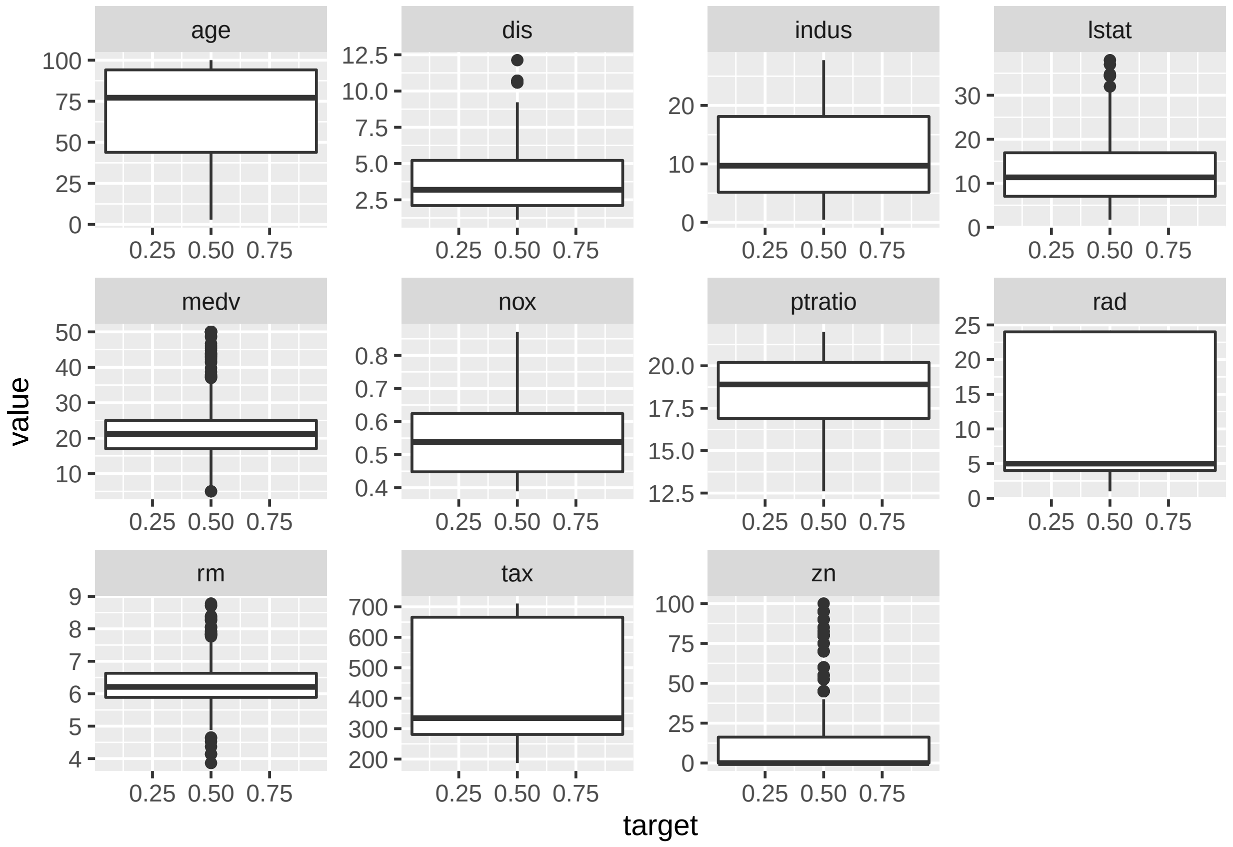


Figure 8. Feature box plots with outliers

Outlier imputation

The boxplots above confirm that there are obvious outliers for variables `dis`, `indus`, `lstat`, `medv`, `ptratio`, `rm`, `tax`, and `zn`. These outliers need to be imputed to prevent them from skewing the results of the modeling **(Code Appendix 3.2.2)**. We use the median values to impute outliers. After imputing the variables with outliers, let's analyze them again with respect to their relation to the `target` variable **(Code Appendix 3.2.3)**. This is to ensure that outliers have been eliminated or at least minimized as much as possible.

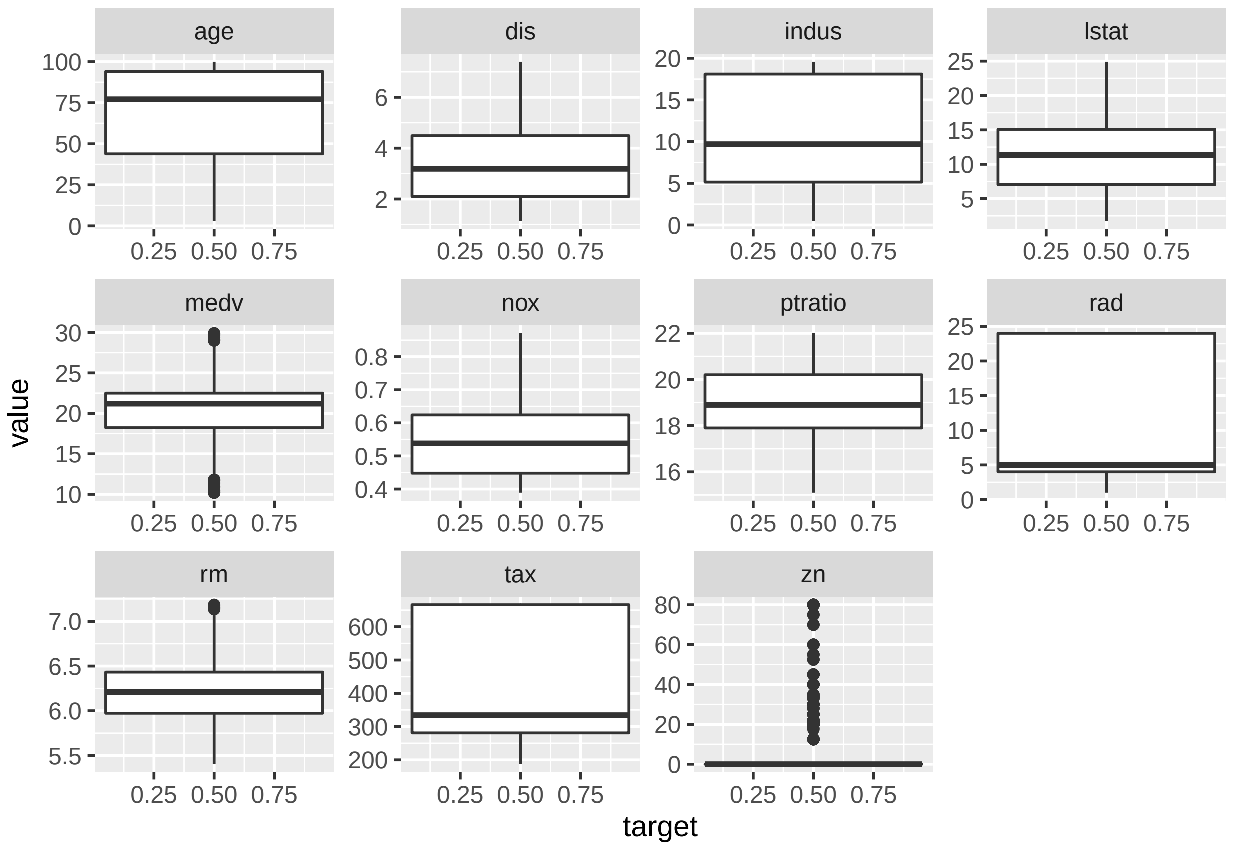


Figure 9. Feature box plots with imputer outliers

Model Building

Model 1: Kitchen Sink

AIC Value = 171.94

This model contains all values except for `tax` which was removed from consideration in any model due to its multicollinearity with other features. No transformations were performed on this model. The rule of thumb for these types are regressions are the AIC value. Lower AIC values indicate a strong model. **(Code Appendix 4.2)**

Model 2: Kitchen Sink Transformed in Part

AIC Value = 173.86

For this model, we include all features, squaring `age` and log transforming `lstat` in hopes of producing a lower AIC score. **(Code Appendix 4.3)**

Model 3: Kitchen Sink Transformed More

AIC Value = 168.1

Now we see the AIC score starting to decrease, we log `zn`, `dis`, and keep `age` squared, though we are unsure if this last transformation has any impact. **(Code Appendix 4.4)**

Model 4: More Transformations

AIC Value = 161.9

This model lowers the AIC score and reduces the residual deviance, logging and scaling `zn`, squaring `age`, and multiplying `rad` by `rm`. **(Code Appendix 4.5)**

Model 5: Variable Importance

AIC Value = 161.9

Keeping model 4's setup, except for deleting `indus` leads to higher residual deviance and the same AIC sore. This could be a final contender for our model. We learn that `indus` and `zn` are not very important to this model. **(Code Appendix 4.6)**

Model 6: Narrow Variable Importance

AIC Value = 161.1

Taking what we've learned from model 5, we delete `indus` and multiply `ptratio` by `nox` and push the AIC even lower. **(Code Appendix 4.7)**

Model Selection

Before proceeding to final model selection, we will test for accuracy and error. As a final check we will look at the ROC plots. **(Code Appendix 5.1)**

Model Error

For each model we use the `predict()` function and check the error and r^2 values (**Code Appendix 5.2.1**). The RMSE is decreasing with each model, possibly meaning that the fit is better with each (**Code Appendix 5.2.2**). It is difficult to tell as the training sample is rather small. Model 6's r^2 value may mean that by deleting `indus` from the model it lost some information that would lower the r^2.

```
##           modelOne  modelTwo modelThree modelFour modelFive  modelSix
## RMSE     0.3071426 0.3018626  0.2945447 0.2917838 0.3000459 0.2928904
## Rsquared 0.6211495 0.6333644  0.6511870 0.6581586 0.6399095 0.6560094
## MAE      0.1726477 0.1714542  0.1597825 0.1515663 0.1550112 0.1514353
```

Confusion Matrix and Accuracy Measurment

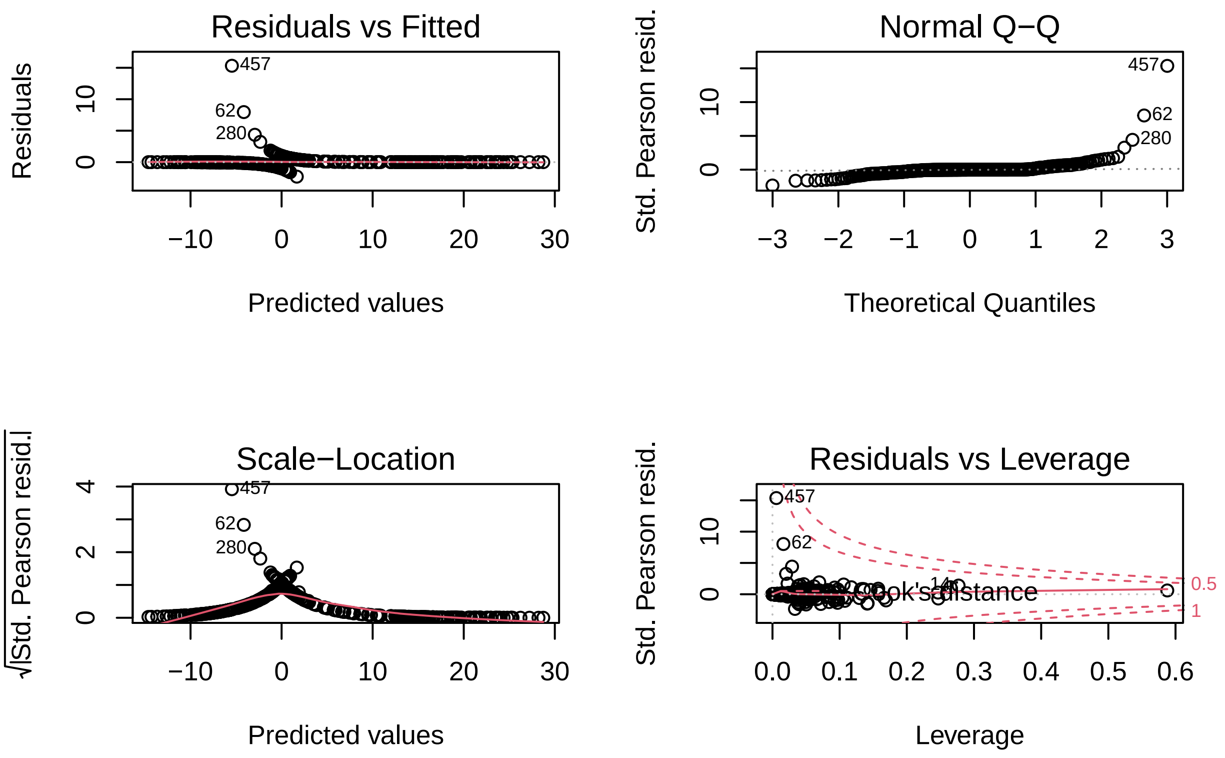
In terms of accuracy, it appears that models 5 and 6 are stand outs with the highest accuracy scores. **(Code Appendix 5.3 & 5.4)**

```
##   cOne.overall cTwo.overall cThree.overall cFour.overall cFive.overall
## 1    0.8817204    0.8709677      0.8817204     0.8924731     0.8817204
##   cSix.overall
## 1    0.8817204
```

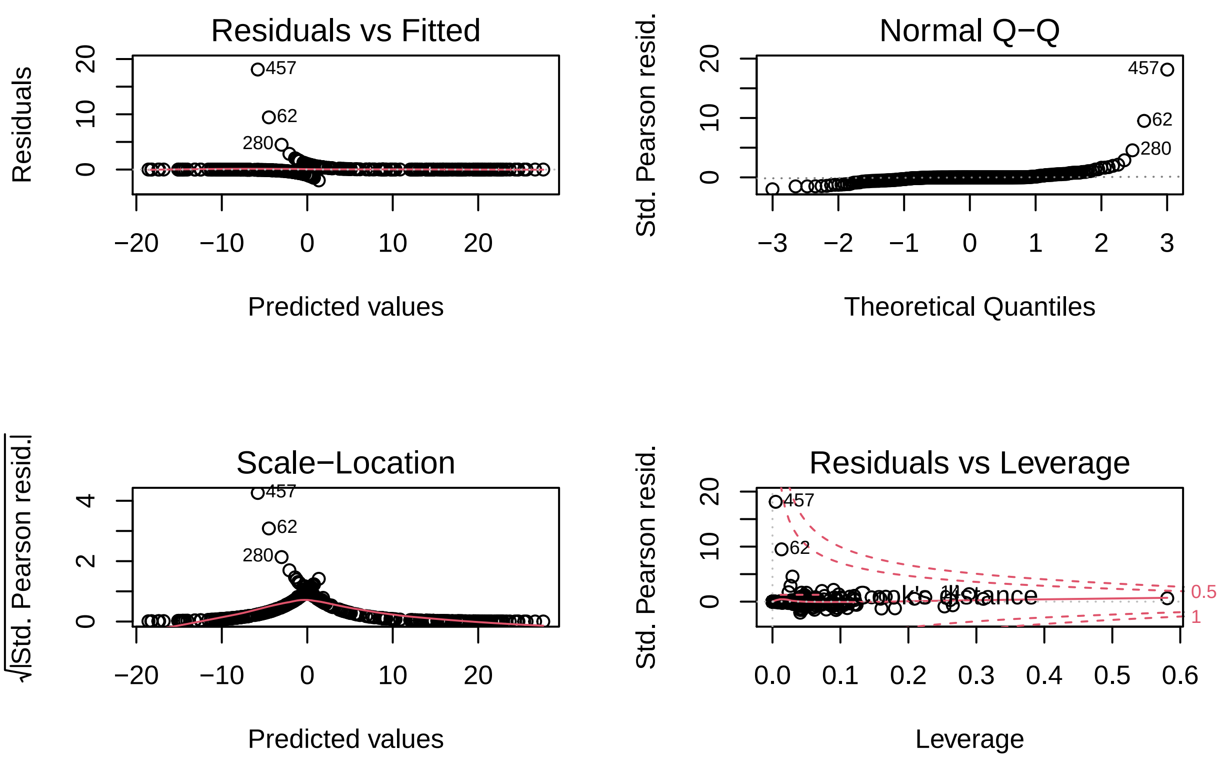
Final Model Plots

In reviewing the final plots for these models, we check again the residuals and QQ plots for each. Both models look extremely close for each of the graphs, with very slight differences. **(Code Appendix 5.5.1 & 5.5.2)**

Model 5



Model 6



ROC

The ROC plot in figure 10 **(Code Appendix 5.6)** seems to provide the tie breaker for us. Model 6 is 0.1 higher and will be our final model to use for our predictions.

```
##
## Call:
## roc.default(response = train$target, predictor = modelFive$fitted.values,     percent = TRUE, plot =
##
## Data: modelFive$fitted.values in 186 controls (train$target 0) < 187 cases (train$target 1).
## Area under the curve: 98.24%
```

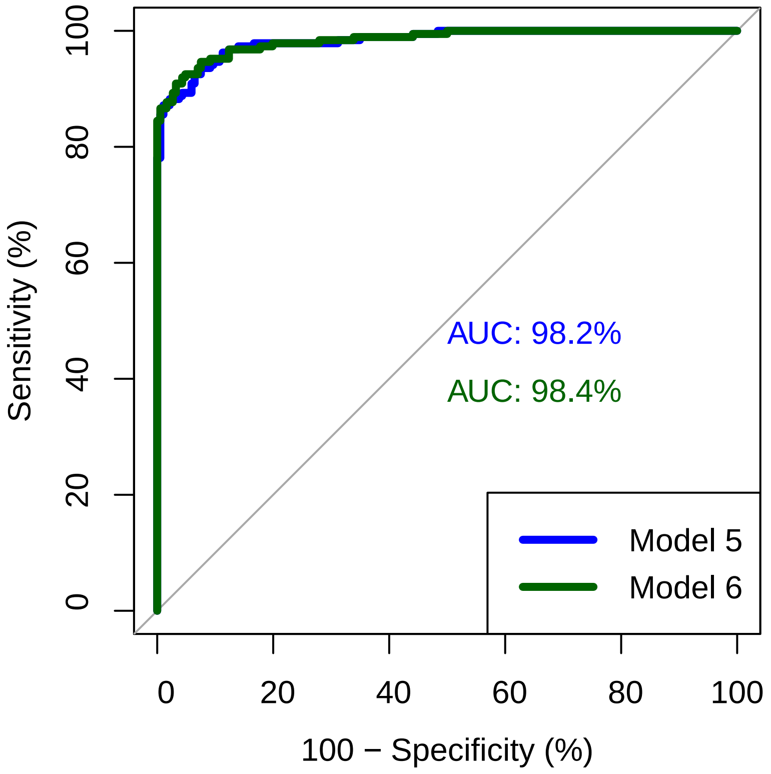


Figure 10. ROC plot with models 5 and 6

Conclusion

We conclude by writing our predictions to the `test` data in a CSV using model 6. **(Code Appendix 5.7)**

Code Appendix

1.1 Libraries Used

We use pretty standard packages for this assignment, including the ever-useful `tidyverse`, `ggplot2`, and `caret`. New additions for this assignment include `VIM`, `DataExplorer`, and `broom`.

{r, warning = FALSE, message = FALSE, echo=FALSE} library(tidyverse) library(ggplot2) library(VIM) library(GGally) library(caret) library(broom) library(kableExtra) library(tidymodels) library(DataExplorer) library(psych) library(pROC)

2.1 Data Exploration

2.2 Data Import

{r, warning = FALSE, message = FALSE, echo=FALSE rawTrain <- read.csv("https://raw.githubusercontent.com/MsQCompSci/Data621Group4/main/HW3/crime-training-data_modified.csv", header = TRUE, stringsAsFactors = FALSE) rawTest <- read.csv("https://raw.githubusercontent.com/MsQCompSci/Data621Group4/main/HW3/crime-evaluation-data_modified.csv")

2.3 Summary Stats

{r, warning = FALSE, message = FALSE, echo=FALSE} glimpse(rawTrain)

2.4 Table 1. Glimpse of data structure**

{r, warning = FALSE, message = FALSE, echo=FALSE} kable(describe(rawTrain),booktabs = TRUE) %>% kable_styling(latex_options = "scale_down")

2.5 Missing Data Checks

2.5.1 Figure 1. Plot missing values with VIM package

{r, warning = FALSE, message = FALSE, echo=FALSE} #plot missing values using VIM package aggr(rawTrain , col=c('navyblue','red'), numbers=TRUE, sortVars=TRUE, labels=names(rawTrain), cex.axis=.7, gap=3, ylab=c("Histogram of missing data","Pattern"))

2.5.2 Figure 2. Plot missing values with DataExplorer package

{r, warning = FALSE, message = FALSE, echo=FALSE} DataExplorer::plot_intro(rawTrain)

2.6 Feature Histograms Figure 3. Feature histograms

{r, warning = FALSE, message = FALSE, echo=FALSE} DataExplorer::plot_histogram(rawTrain)

2.7 Feature Boxplots Figure 4. Feature box plots

{r, warning = FALSE, message = FALSE, echo=FALSE} ggplot(stack(rawTrain), aes(x = ind, y = values)) + geom_boxplot(color = "darkblue", fill = "lightblue", alpha = 0.2, outlier.color = "red", outlier.fill = "red", outlier.alpha = 0.2, notch = TRUE) + labs(title = "Boxplot of all feature variables") + scale_y_log10()

2.8 Feature QQ Plots

2.8.1 QQ Plots Figure 5. Feature QQ plot

{r, warning = FALSE, message = FALSE, echo=FALSE} qq_train_data <- rawTrain[, c("age", "dis", "indus", "lstat", "medv", "nox", "ptratio", "rad", "rm", "tax", "zn")]

DataExplorer::plot_qq(qq_train_data, nrow = 4L, ncol = 3L)

2.8.2 Log QQ Plots Figure 6. Feature QQ plots, log transformation

{r, warning = FALSE, message = FALSE, echo=FALSE} log_qq_train_data <- DataExplorer::update_columns(qq_train_d
ind = names(qq_train_data), what = log)

DataExplorer::plot_qq(log_qq_train_data, nrow = 4L, ncol = 3L)

2.9 Correlation Figure 7. Correlation matrix

{r, warning = FALSE, message = FALSE, echo=FALSE} #correlation matrix for predictors ggcorr(rawTrain%>% select(zn:medv))

3.1 Data Preparation

3.2 Outlier Value Analysis

3.2.1 Figure 8. Feature box plots with outliers

{r, warning = FALSE, message = FALSE, echo=FALSE} pred_vs_target <- gather(rawTrain, variable, value, -c(chas,target))

ggplot(pred_vs_target, aes(x = target, y = value)) + geom_boxplot() + facet_wrap(~variable, scales = 'free')

3.2.2 Outlier imputation

{r, warning = FALSE, message = FALSE, echo=FALSE} rawTrain_preped <- rawTrain %>% mutate(dis = ifelse(dis > 7.5, median(dis), dis), indus = ifelse(indus > 21, median(indus), indus), lstat = ifelse(lstat > 25, median(lstat), lstat), medv = ifelse(medv > 30 | medv < 10, median(medv), medv), ptratio = ifelse(ptratio < 15.0, median(ptratio), ptratio), rm = ifelse(rm > 7.2 | rm < 5.4, median(rm), rm), tax = ifelse(tax > 700.0, median(tax), tax), zn = ifelse(zn > 80, median(zn), zn))

3.2.3 Figure 9. Feature box plots with imputed outliers

{r, warning = FALSE, message = FALSE, echo=FALSE} pred_vs_target <- gather(rawTrain_preped, variable, value, -c(chas,target))

ggplot(pred_vs_target, aes(x = target, y = value)) + geom_boxplot() + facet_wrap(~variable, scales = 'free')

4.1 Model Building

{r, warning = FALSE, message = FALSE, echo=FALSE} dt <- createDataPartition(rawTrain_preped$target, p = .8, list = FALSE, times = 1)

train <- rawTrain[dt,] test <- rawTrain[-dt,]

4.2 Model 1: Kitchen Sink

```
#remove Tax due to high correlation with other variables
modelOne <- glm(target ~ zn + indus + chas + nox + rm + age + dis + rad + ptratio + lstat + medv , data
summary(modelOne)
```

```
##
## Call:
## glm(formula = target ~ zn + indus + chas + nox + rm + age + dis +
##     rad + ptratio + lstat + medv, family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.69325  -0.10390   0.00002   0.00425   2.82842
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -46.49997    8.05045  -5.776 7.65e-09 ***
## zn           -0.09361    0.04362  -2.146  0.03185 *
## indus        -0.09750    0.05077  -1.921  0.05479 .
## chas          0.95973    0.83737   1.146  0.25175
## nox          54.73869    9.56435   5.723 1.05e-08 ***
## rm           -1.01428    0.80635  -1.258  0.20844
## age           0.04889    0.01667   2.933  0.00336 **
## dis           1.25328    0.30638   4.091 4.30e-05 ***
## rad           0.50049    0.16936   2.955  0.00312 **
## ptratio       0.34328    0.14059   2.442  0.01462 *
## lstat         0.08076    0.06272   1.288  0.19785
## medv          0.27529    0.08276   3.326  0.00088 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 517.09  on 372  degrees of freedom
## Residual deviance: 140.84  on 361  degrees of freedom
## AIC: 164.84
##
## Number of Fisher Scoring iterations: 9
```

4.3 Model 2: Kitchen Sink Transformed in Part

```r
#remove Tax squared age and log lstat
modelTwo <- glm(target ~ zn + indus + chas + nox + rm + age^2 + dis + rad + ptratio + log2(lstat) + medv
summary(modelTwo)
```

```
##
## Call:
## glm(formula = target ~ zn + indus + chas + nox + rm + age^2 +
##     dis + rad + ptratio + log2(lstat) + medv, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6962  -0.1133   0.0000   0.0053   3.2319
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -45.45718    8.50945  -5.342 9.19e-08 ***
## zn           -0.08659    0.04165  -2.079 0.037644 *
## indus        -0.08868    0.05035  -1.761 0.078210 .
## chas          1.05977    0.85375   1.241 0.214492
## nox          54.43119    9.50437   5.727 1.02e-08 ***
## rm           -1.22175    0.80919  -1.510 0.131084
## age           0.05344    0.01656   3.227 0.001252 **
## dis           1.24031    0.30327   4.090 4.32e-05 ***
## rad           0.49904    0.16525   3.020 0.002528 **
## ptratio       0.36544    0.14032   2.604 0.009202 **
## log2(lstat)   0.19775    0.55668   0.355 0.722417
## medv          0.27180    0.08175   3.325 0.000885 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 517.09  on 372  degrees of freedom
## Residual deviance: 142.39  on 361  degrees of freedom
## AIC: 166.39
##
## Number of Fisher Scoring iterations: 9
```

```
#This one has a litter lower AIC
```

4.4 Model 3: Kitchen Sink Transformed More

```
#log10(zn + 1), log10(dis) and deleted log2(lstat) - not significant
modelThree <- glm(target ~ log10(zn + 1) + indus + chas + nox + rm + age^2 + log10(dis) + rad + ptratio
summary(modelThree)
```

```
##
## Call:
## glm(formula = target ~ log10(zn + 1) + indus + chas + nox + rm +
##     age^2 + log10(dis) + rad + ptratio + medv, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8027  -0.1228   0.0000   0.0036   3.3613
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -51.58699    8.88099  -5.809 6.30e-09 ***
## log10(zn + 1)  -1.02160    0.59339  -1.722 0.085136 .
## indus          -0.02894    0.05251  -0.551 0.581593
## chas            0.70699    0.86123   0.821 0.411704
## nox            59.71717    9.83889   6.070 1.28e-09 ***
## rm             -1.55523    0.80498  -1.932 0.053360 .
## age             0.06316    0.01684   3.751 0.000176 ***
## log10(dis)     13.68072    2.98890   4.577 4.71e-06 ***
## rad              0.57489    0.17883   3.215 0.001305 **
## ptratio          0.40897    0.15396   2.656 0.007897 **
## medv             0.31003    0.08719   3.556 0.000377 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 517.09  on 372  degrees of freedom
## Residual deviance: 135.89  on 362  degrees of freedom
## AIC: 157.89
##
## Number of Fisher Scoring iterations: 9
```

```
#AIC is lower again (not sure if age^2 ishelpful)
```

4.5 Model 4: More Transformations

```
#combine rad and rm (multiplied) - they seemed to correspond in their distributions
modelFour<- glm(target ~ log10(zn + 1) + indus + chas + nox +  age^2 + log10(dis) + rad*rm + ptratio  +
summary(modelFour)
```

```
##
## Call:
## glm(formula = target ~ log10(zn + 1) + indus + chas + nox + age^2 +
##     log10(dis) + rad * rm + ptratio + medv, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7055  -0.0867   0.0000   0.0067   3.3484
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -41.56772    8.74709  -4.752 2.01e-06 ***
## log10(zn + 1)  -1.06233    0.64614  -1.644 0.100149
## indus          -0.04292    0.05323  -0.806 0.420105
## chas            0.38535    0.91770   0.420 0.674556
## nox            63.28481   10.40924   6.080 1.20e-09 ***
## age             0.07134    0.01832   3.893 9.90e-05 ***
## log10(dis)     14.22699    3.13029   4.545 5.50e-06 ***
## rad            -1.65250    0.58813  -2.810 0.004958 **
## rm             -3.99559    1.28177  -3.117 0.001825 **
## ptratio         0.45632    0.16007   2.851 0.004362 **
## medv            0.36106    0.09629   3.750 0.000177 ***
## rad:rm          0.37283    0.11729   3.179 0.001479 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 517.09  on 372  degrees of freedom
## Residual deviance: 128.01  on 361  degrees of freedom
## AIC: 152.01
##
## Number of Fisher Scoring iterations: 9
```

```
#AIC is lower #Not sure what the rationale is for this working but it lowered the AIC numnber and Resid
```

4.6 Model 5: Variable Importance

```
#delete indus
modelFive<-glm(target ~ log10(zn+1)+ nox +  age^2 + log10(dis) + rad*rm + ptratio  + medv, data = train
summary(modelFive)
```

```
##
## Call:
## glm(formula = target ~ log10(zn + 1) + nox + age^2 + log10(dis) +
##     rad * rm + ptratio + medv, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9188  -0.0840   0.0000   0.0062   3.3053
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -40.84765    8.59008  -4.755 1.98e-06 ***
## log10(zn + 1)  -1.16391    0.61252  -1.900 0.057408 .
## nox            61.80810   10.28136   6.012 1.84e-09 ***
## age             0.07310    0.01784   4.098 4.17e-05 ***
## log10(dis)     14.98689    3.05894   4.899 9.61e-07 ***
## rad            -1.69601    0.58562  -2.896 0.003779 **
## rm             -4.16888    1.26172  -3.304 0.000953 ***
## ptratio         0.43496    0.15989   2.720 0.006521 **
## medv            0.38187    0.09471   4.032 5.53e-05 ***
## rad:rm          0.38523    0.11608   3.319 0.000904 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 517.09  on 372  degrees of freedom
## Residual deviance: 128.74  on 363  degrees of freedom
## AIC: 148.74
##
## Number of Fisher Scoring iterations: 9
```

```
#AIC is higher #resiudal deviance is lower
# I looked at the histograms and looked for complementary shapes to decide what to multiply
```

4.7 Model 6: Narrow Variable Importance

```
#multiply ptratio*nox (remove squared from age)
modelSix<- glm(target ~ log10(zn + 1) + age  + ptratio*nox + log10(dis) + rad*rm + medv, data = train, 
summary(modelSix)
```

```
##
## Call:
## glm(formula = target ~ log10(zn + 1) + age + ptratio * nox +
##     log10(dis) + rad * rm + medv, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7766  -0.0768   0.0000   0.0062   3.4043
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -95.71180   29.57503  -3.236  0.00121 **
## log10(zn + 1)  -1.34704    0.65374  -2.061  0.03935 *
## age             0.07730    0.01877   4.118 3.82e-05 ***
## ptratio         3.36184    1.46694   2.292  0.02192 *
## nox           155.10748   49.43191   3.138  0.00170 **
## log10(dis)     14.53015    3.08560   4.709 2.49e-06 ***
## rad            -1.60795    0.58100  -2.768  0.00565 **
## rm             -4.02658    1.28176  -3.141  0.00168 **
## medv            0.38772    0.09850   3.936 8.28e-05 ***
## ptratio:nox    -5.09404    2.53095  -2.013  0.04415 *
## rad:rm          0.37374    0.11554   3.235  0.00122 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 517.09  on 372  degrees of freedom
## Residual deviance: 123.86  on 362  degrees of freedom
## AIC: 145.86
##
## Number of Fisher Scoring iterations: 9
```

```
#AIC is lower
```

5.1 Model Selection

Before proceeding to final model selection, we will test for accuracy and error. As a final check we will look at the ROC plots.

5.2 Model Error

5.2.1

{r, warning = FALSE, message = FALSE} #Make predictions predOne = predict(modelOne,test, type = “response”) predTwo = predict(modelTwo,test, type = “response”) predThree = predict(modelThree,test, type = “response”) predFour = predict(modelFour,test, type = “response”) predFive = predict(modelFive,test, type = “response”) predSix = predict(modelSix,test, type = “response”)

##5.2.2

{r, warning = FALSE, message = FALSE} #Error Measures data.frame(modelOne = postResample(pred = predOne, obs = test$target), modelTwo = postResample(pred = predTwo, obs = test$target), modelThree = postResample(pred = predThree, obs = test$target), modelFour = postResample(pred = predFour, obs = test$target), modelFive = postResample(pred = predFive, obs = test$target), modelSix = postResample(pred = predSix, obs = test$target))

5.3 Confusion Matrix and Accuracy Measurment

{r, warning = FALSE, message = FALSE, echo=FALSE} Extract Accuracy

5.3.1 Model 1

resultsFitOne <- ifelse(predOne > 0.5,1,0) resultsFitOne <- as.factor(resultsFitOne)

cOne <- confusionMatrix(as.factor(test$target), resultsFitOne) accOne < -as.data.frame(cOne$overall)[1] accOne<- accOne %>% slice(1)

5.3.2 Model 2

resultsFitTwo <- ifelse(predTwo > 0.5,1,0) resultsFitTwo <- as.factor(resultsFitTwo)

cTwo <- confusionMatrix(resultsFitTwo, as.factor(test$target)) accTwo < -as.data.frame(cTwo$overall)[1] accTwo<- accTwo %>% slice(1)

5.3.3 Model 3

resultsFitThree<- ifelse(predThree > 0.5,1,0) resultsFitThree <- as.factor(resultsFitThree)

cThree <- confusionMatrix(resultsFitThree, as.factor(test$target)) accThree < -as.data.frame(cThree$overall)[1] accThree<- accThree%>% slice(1)

5.3.4 Model 4

resultsFitFour<- ifelse(predFour > 0.5,1,0) resultsFitFour <- as.factor(resultsFitFour)

cFour <- confusionMatrix(resultsFitFour, as.factor(test$target)) accFour < -as.data.frame(cFour$overall)[1] accFour<- accFour%>% slice(1)

5.3.5 Model 5

resultsFitFive<- ifelse(predFive > 0.5,1,0) resultsFitFive <- as.factor(resultsFitFive)

cFive <- confusionMatrix(resultsFitFive, as.factor(test$target)) accFive < -as.data.frame(cFive$overall)[1] accFive<- accFive%>% slice(1)

5.3.6 Model 6

resultsFitSix<- ifelse(predSix > 0.5,1,0) resultsFitSix <- as.factor(resultsFitSix)

cSix<- confusionMatrix(resultsFitSix, as.factor(test$target))accSix < -as.data.frame(cSix$overall)[1] accSix<- accSix%>% slice(1)

5.4 Accuracy Results

{r, warning = FALSE, message = FALSE, echo=FALSE} #create a table with accuracies data.frame(c(accOne, accTwo, accThree, accFour,accFive, accSix))

5.5 Final Model Plots

5.5.1 Model 5

{r, warning = FALSE, message = FALSE, echo=FALSE} par(mfrow = c(2,2)) plot(modelFive)

5.5.2 Model 6

{r, warning = FALSE, message = FALSE, echo=FALSE} par(mfrow = c(2,2)) plot(modelSix)

5.6 ROC Figure 10. ROC plot with models 5 and 6

{r, warning = FALSE, message = FALSE, echo=FALSE} par(pty = "s") roc(train$target, modelFive$fitted.values, plot = TRUE, legacy.axes = TRUE, percent=TRUE, col="blue", lwd=4, print.auc = TRUE) plot.roc(train$target, modelSix$fitted.values, percent=TRUE, col="dark green", lwd=4, print.auc=TRUE, add=TRUE, print.auc.y=40) legend("bottomright", legend=c("Model 5", "Model 6"), col=c("blue", "dark green"), lwd=4)

5.7 Conclusion

{r echo=FALSE, message=FALSE, warning=FALSE} rawTest$target_pred <- predict(modelSix, newdata = rawTest) rawTest %>% write.csv(., "crime.csv", row.names = F) "'