DATA 621—Assignment no. 3

Critical Thinking Group 2

October 30, 2019

Contents

Executive Overview

blah balh

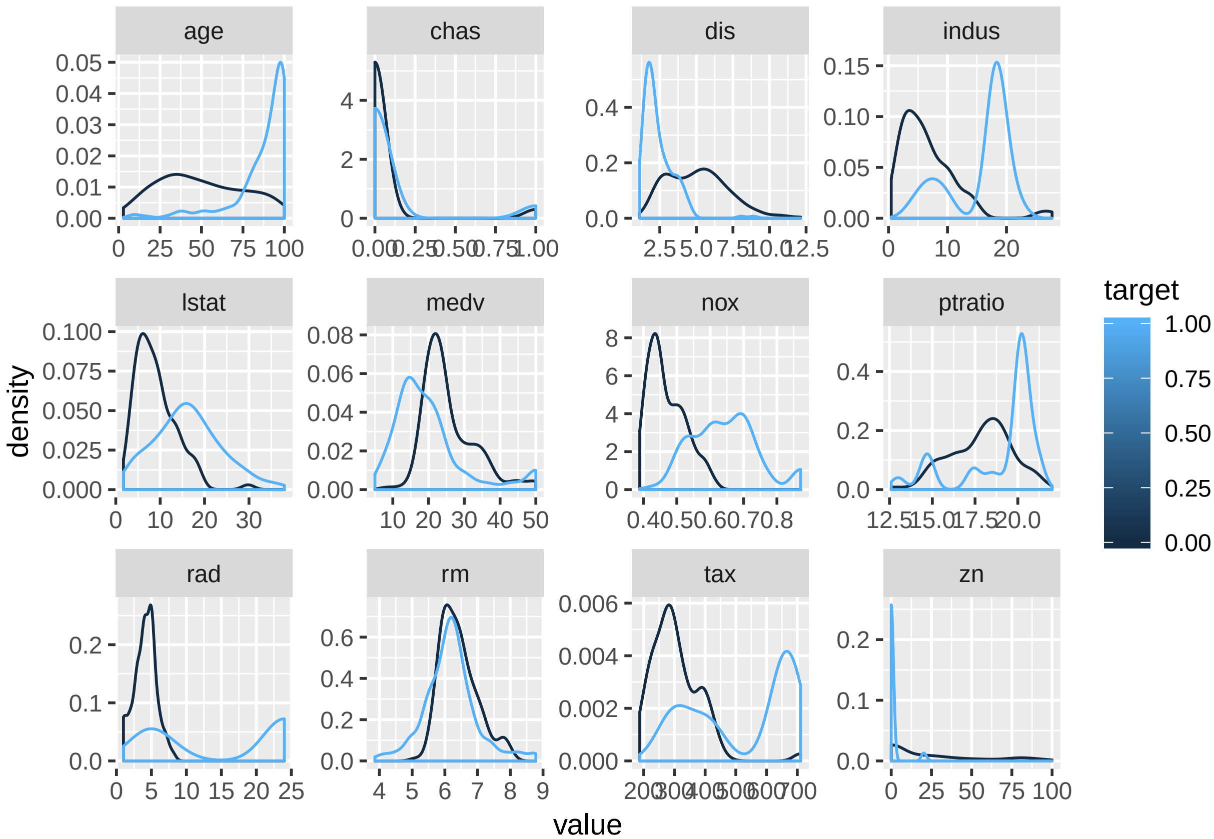
Create train and test sets using the `caret` machine learning package:

Only use the `train` data frame until the very end of the process, when we use test to evaluate how effective the model is!

Data Exploration

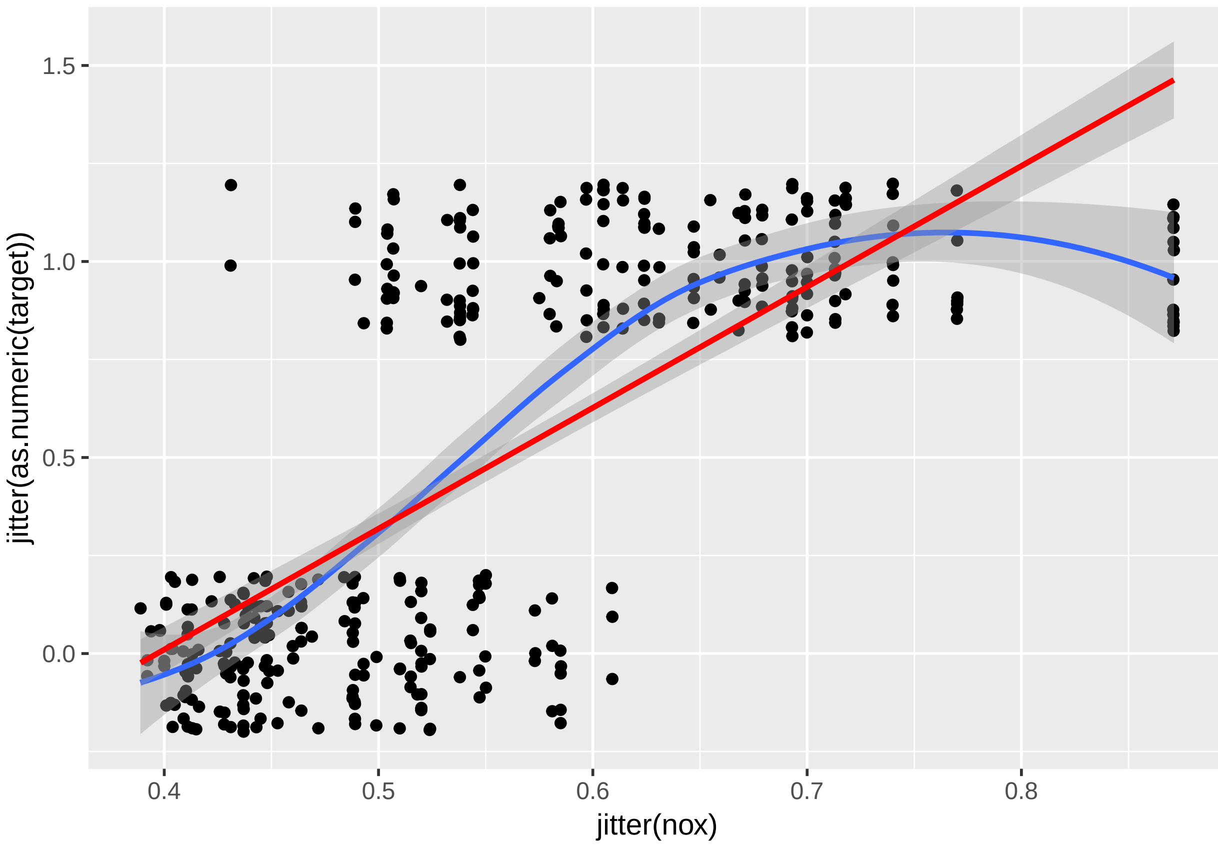
There don't seem to be any outliers or missing data, so we will proceed directly to examining the variables. First, histograms of each variable for each `target` class:

```
train %>%
  gather(-target, key='variable', value='value') %>%
  ggplot(aes(x=value, group=target, color=target)) +
    facet_wrap(~ variable, scales='free') +
    geom_density()
```



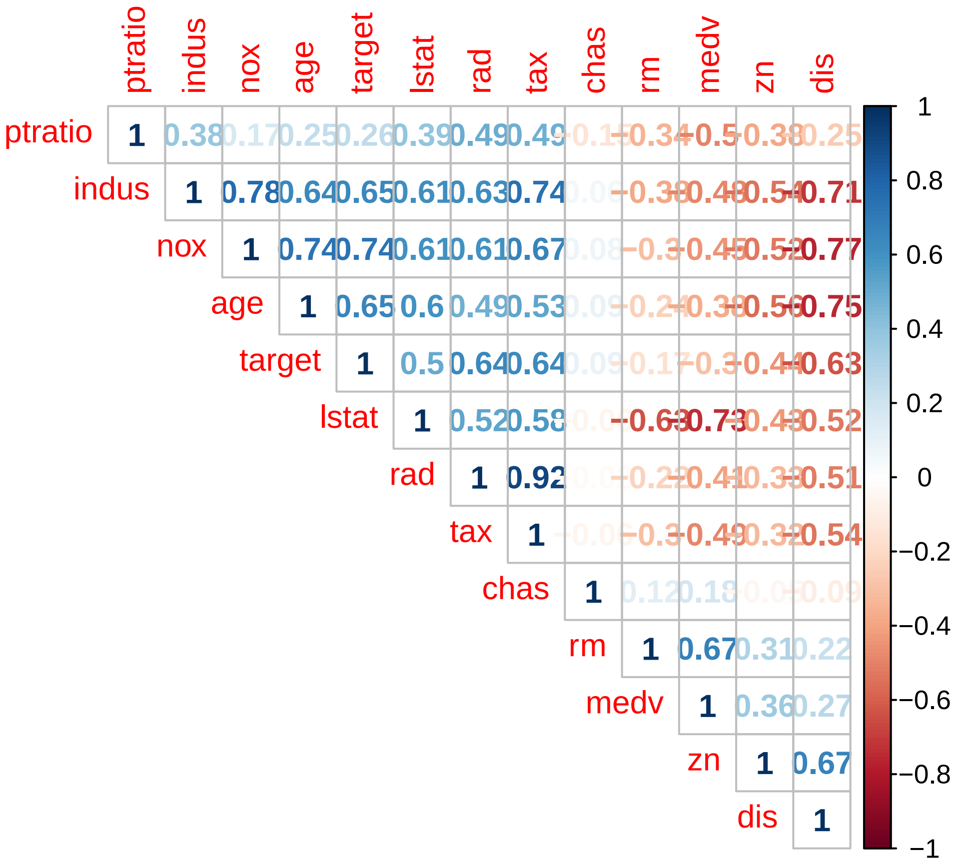
Most variables have distinct shapes for each `target` class. `chas` and `zn` are quite skewed, and do not appear terribly informative. `indus` and `tax` have two peaks for `target = 1`, indicating there are two seperate processes at work there.

```
ggplot(train, aes(x=jitter(nox), y=jitter(as.numeric(target)))) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method='lm', color='red')
```



It is to be expected that many of these variables will be correlated with each other:

```
corrplot(cor(train), type='upper', method='number', order='hclust')
```



Obviously, the concentration of industry is strongly and positively correlated with nitrogen oxide concentration $\rho = 0.78$). Parent-teacher ratio is negatively correlated with median property values ($\rho = -0.5$), and positively correlated with property taxes ($\rho = 0.49$). What these and other variables are really getting at is *economic class*. Each measures a different phenomenon, but can be conceived of as operationalizing one thing. This suggests PCA may be useful on this dataset.

Checking for interactions

Given the high correlation between the variables, it may be the case that there are numerous interactions that can improve our modeling. In this section, we attempt to determine if this is the case. We will group numeric variables by membership in quartile, and examine line plots.

```
calc_percentile <- function(x){
  trunc(rank(x)) / length(x)
}
```

Data Preparation

Modeling

Function to calculate McFadden's pseudo-R^2 for logistic models:

```r
calc_r2 <- function(model) {
  1 - model$deviance / model$null.deviance
}
```

M_0: Dummy model

Baseline model, which just predicts the class proportion, which is nearly balanced between the two classes. If we are having trouble improving on this model, we know we are doing something wrong.

This dummy model has an accuracy of about 0.50, sensitivity of 1, and specificity of 0. Since it has zero predictive power, we know that it has a pseudo-R^2 of 0.

```r
m_0 <- glm(target ~ 1, train, family=binomial())
pred_0 <- factor(round(predict(m_0, train, type='response')), levels=c('0', '1'))
confusionMatrix(data=pred_0, reference=factor(train$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 187 186
##          1   0   0
##
##                Accuracy : 0.5013
##                  95% CI : (0.4494, 0.5532)
##     No Information Rate : 0.5013
##     P-Value [Acc > NIR] : 0.5207
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.5013
##          Neg Pred Value :    NaN
##              Prevalence : 0.5013
##          Detection Rate : 0.5013
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

M_1: Full model

The next simplest model uses all available data, without transformations or interactions or polynomials:

```r
m_1 <- glm(target ~ zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + lstat + medv, trai
pred_1 <- factor(round(predict(m_1, train, type='response')), levels=c('0', '1'))
calc_r2(m_1)
```

```
## [1] 0.7216759
```

```r
confusionMatrix(data=pred_1, reference=factor(train$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 173  19
##          1  14 167
##
##                Accuracy : 0.9115
##                  95% CI : (0.878, 0.9383)
##     No Information Rate : 0.5013
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.823
##
##  Mcnemar's Test P-Value : 0.4862
##
##             Sensitivity : 0.9251
##             Specificity : 0.8978
##          Pos Pred Value : 0.9010
##          Neg Pred Value : 0.9227
##              Prevalence : 0.5013
##          Detection Rate : 0.4638
##    Detection Prevalence : 0.5147
##       Balanced Accuracy : 0.9115
##
##        'Positive' Class : 0
##
```

M_2: Stepwise variable selection with interactions

We know that variable interaction is probably likely. We can automatically test all interactions using stepwise selection:

```r
m_2 <- stepAIC(m_1, trace=0, scope=list(upper = ~zn*indus*chas*nox*rm*age*dis*rad*tax*ptratio*lstat*medv
summary(m_2)
```

```
##
## Call:
## glm(formula = target ~ zn + indus + chas + nox + rm + age + dis +
##     rad + tax + ptratio + lstat + medv + ptratio:lstat + chas:tax +
##     nox:age + rm:lstat + rm:age + age:medv + nox:ptratio + dis:tax +
##     indus:tax + tax:medv + indus:dis + age:lstat, family = binomial(),
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.70636  -0.00332   0.00000   0.00000   2.57482
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    2.503e+00  8.456e+01   0.030 0.976389
```

```
## zn           -4.539e-01  1.865e-01  -2.433 0.014958 *
## indus        -2.261e+00  8.363e-01  -2.703 0.006862 **
## chas         -6.926e+03  4.370e+05  -0.016 0.987355
## nox           3.607e+02  1.952e+02   1.848 0.064661 .
## rm           -2.264e+01  7.012e+00  -3.228 0.001247 **
## age          -2.219e+00  5.930e-01  -3.742 0.000182 ***
## dis          -1.474e+01  5.996e+00  -2.459 0.013933 *
## rad           2.495e+00  6.857e-01   3.639 0.000274 ***
## tax          -3.465e-01  1.186e-01  -2.922 0.003473 **
## ptratio       8.824e+00  4.858e+00   1.817 0.069279 .
## lstat        -1.399e+00  2.456e+00  -0.570 0.568786
## medv          9.272e-01  7.603e-01   1.220 0.222642
## ptratio:lstat  2.242e-01  1.271e-01   1.764 0.077731 .
## chas:tax      2.502e+01  1.578e+03   0.016 0.987343
## nox:age       1.362e+00  5.341e-01   2.549 0.010789 *
## rm:lstat     -5.908e-01  2.796e-01  -2.113 0.034585 *
## rm:age        3.657e-01  9.577e-02   3.819 0.000134 ***
## age:medv     -3.075e-02  9.117e-03  -3.372 0.000745 ***
## nox:ptratio  -1.843e+01  9.957e+00  -1.851 0.064239 .
## dis:tax       5.026e-02  1.892e-02   2.656 0.007913 **
## indus:tax     5.110e-03  1.884e-03   2.713 0.006672 **
## tax:medv      4.637e-03  1.871e-03   2.477 0.013231 *
## indus:dis     1.913e-01  1.284e-01   1.490 0.136209
## age:lstat     7.722e-03  3.927e-03   1.966 0.049257 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 517.085  on 372  degrees of freedom
## Residual deviance:  50.534  on 348  degrees of freedom
## AIC: 100.53
##
## Number of Fisher Scoring iterations: 25
```

However, this model is probably overfit. By common heuristic, we have enough data for:

```
min(table(train$target)) / 15
```

```
## [1] 12.4
```

i.e., 12 variables.

M_3: Adjusting for multiple significance tests

To correct for this overfitting, we will use the `p.adjust` function to revise our p-values, and then use those that remain significant at $p = 0.05$ for the next model:

```
m_2_p <- summary(m_2)$coefficients[,4]
sort(p.adjust(m_2_p))
```

```
##       rm:age          age          rad     age:medv           rm
##  0.003352611  0.004375544  0.006294925  0.016389911  0.026180977
##          tax        indus    indus:tax      dis:tax      nox:age
##  0.069463666  0.126774120  0.126774120  0.134529104  0.172628530
```

```
##            zn           dis      tax:medv      rm:lstat     age:lstat
##   0.198464177   0.198464177   0.198464177   0.415017379   0.541826029
##           nox       ptratio ptratio:lstat   nox:ptratio     indus:dis
##   0.642394932   0.642394932   0.642394932   0.642394932   0.817253780
##   (Intercept)          chas         lstat          medv      chas:tax
##   1.000000000   1.000000000   1.000000000   1.000000000   1.000000000
```

Using the top values:

```r
m_3 <- glm(target ~ rm:age + age + rad + age:medv + rm, train, family=binomial())
pred_3 <- factor(round(predict(m_3, train, type='response')), levels=c('0', '1'))
confusionMatrix(data=pred_3, reference=factor(train$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 156  22
##          1  31 164
##
##                Accuracy : 0.8579
##                  95% CI : (0.8183, 0.8917)
##     No Information Rate : 0.5013
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7159
##
##  Mcnemar's Test P-Value : 0.2718
##
##             Sensitivity : 0.8342
##             Specificity : 0.8817
##          Pos Pred Value : 0.8764
##          Neg Pred Value : 0.8410
##              Prevalence : 0.5013
##          Detection Rate : 0.4182
##    Detection Prevalence : 0.4772
##       Balanced Accuracy : 0.8580
##
##        'Positive' Class : 0
##
```

```r
calc_r2(m_3)
```

```
## [1] 0.5679575
```

The psuedo-R^2 is naturally much less than the overfit M_2. Presumably, it will be better fit to the hold-out sample, however. We do see theat sensitivity, specificity, and pos/neg predictive value are actually still pretty strong.

M_4: Previous model + a few more predictors

We noted above that we have data for up to 12 variables in this model, so I will include the first 12 significant variables of the p-value adjustment:

```r
m_4 <- glm(target ~ rm:age + age + rad + age:medv + rm + tax + indus +
             indus:tax + dis:tax + nox:age + zn + dis, train, family=binomial())
pred_4 <- factor(round(predict(m_4, train, type='response')), levels=c('0', '1'))
confusionMatrix(data=pred_4, reference=factor(train$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 180  15
##          1   7 171
##
##                Accuracy : 0.941
##                  95% CI : (0.9121, 0.9627)
##     No Information Rate : 0.5013
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.882
##
##  Mcnemar's Test P-Value : 0.1356
##
##             Sensitivity : 0.9626
##             Specificity : 0.9194
##          Pos Pred Value : 0.9231
##          Neg Pred Value : 0.9607
##              Prevalence : 0.5013
##          Detection Rate : 0.4826
##    Detection Prevalence : 0.5228
##       Balanced Accuracy : 0.9410
##
##        'Positive' Class : 0
##
```

```r
calc_r2(m_4)
```

```
## [1] 0.7321437
```

Despite adding all these variables, and their significance, we see that the confusion matrix evaluations are not that much higher. Psuedo-R^2 did take a nice bump, though. Nonetheless, it is possible that this model does not fit the hold out sample as well as M_3.

Evaluating the Models on the Test Set

```r
# Don't run until the very end
# confusionMatrix(data=predict(model, test), reference=test$target)
```